ホームサービスロボットのための基盤システムの確立：知的処理の実装および組み込みアクセラレータの統合

博士論文

ホームサービスロボットのための基盤システムの確立: 知的処理の実装および組み込みアクセラレータの統合

九州工業大学 大学院 生命体工学研究科 生命体工学専攻
田向研究室

学籍番号 17899004
石田 裕太郎

目次

第 1 章

序論

1.1　研究背景

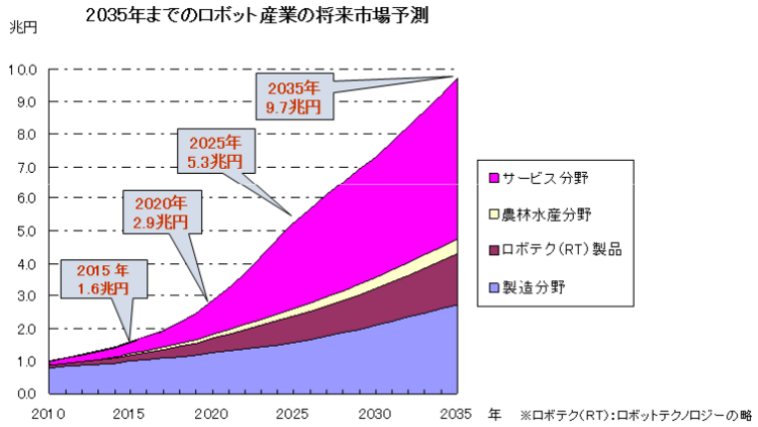


Fig. 1.1　ロボット産業の市場予測 [1]

近年，少子高齢化などの社会問題を背景に，ホームサービスロボットが注目を浴びている．Fig. 1.1 に示す経済産業省・NEDO（新エネルギー・産業技術総合研究機構）の 2035 年に向けたロボット産業の市場予測 [1] によると，ロボットの市場規模は 2015 年約 1.6 兆円から，2035 年約 9.7 兆円へと約 6 倍も成長すると予測されている．その中でもホームサービスロボットを含むサービスロボット分野の市場規模は，2015 年約 3700 億円から 2035 年約 4 兆 9000 億円へと 13 倍もの成長が予測されており，世間の期待がうかがえる．

ホームサービスロボットは，家庭・公共空間で人と共存し，手助けを行う．例えば，子供部屋のおもちゃを片付けたり，レストランのウェイターとして働くことが期待されている．このように働くため，ホームサービスロボットの知的処理は大きく，認識・判断・制御，3 つの機能から成り立っている．認識は，センサデータを信号処理し，符号化を行う．判断は，複数の符号を包括的に考慮し，ロボットの最適な行動を導く．制御は，行動を実現できる計画をたて，アクチュエータを操作する．

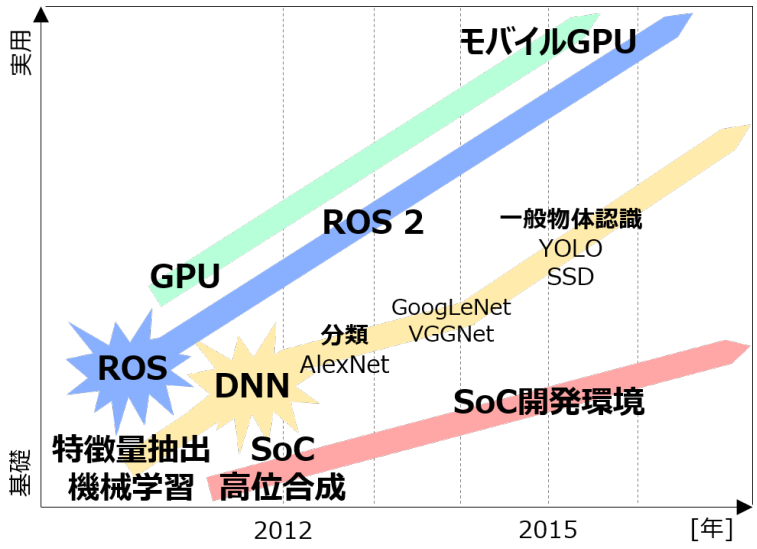


Fig. 1.2　ホームサービスロボットに関連する技術の推移

Fig. 1.2 に，ホームサービスロボットに関連する技術の推移を示す．Fig. 1.2 の横方向は年代を表し，縦方向はホームサービスロボットにおける実用性を表す．Fig. 1.2 には 4 つの技術が図示されており，青色はロボットのソフトウェア，黄色は画像処理，緑色は GPU（Graphic Processing Unit），赤色は FPGA（Field Programmable Gate Array）をそれぞれ示す．

ロボットのソフトウェアとして，ROS（Robot Operating System）が 2007 年に誕生し，2009 年に発表された [2]．ROS は，通信，ツール群，機能群，エコシステムをロボットのソフトウェアにもたらし，リソースの再利用性を大幅に高めた．また，2014 年からは ROS 2 の開発が始まり，主に組み込み・実時間システムへの対応が進み，プロトタイプから製品展開まで応用の場が広まりつつある [3]．そのためロボット分野では，1 から独自規格を用いてソフトウェアを実装することは減り，ロボットミ

ドルウェアを用いた実装がデファクトスタンダードになりつつある．なかでも自動運転においては，ROS をベースとした OSS（Open Source Software）の Autoware[4] が誕生し，自動運転の研究開発のための基盤ソフトウェアとなりつつある．

画像処理はホームサービスロボットの知的処理のうち認識に密接に関わる．特に，ロボットが日用品・雑貨・家事道具を操作して働くためには，人間が目で見るように，カメラの画像から物体を検出・認識する技術が重要である．ここで，物体の検出とは画像のどの領域に物体が存在するか見つけることであり，認識とは検出した物体がどのカテゴリに属するか，すなわち何なのか判別することである．2012 年以前は，物体を認識する手法として，画像から HOG（Histogram of Oriented Gradients）[5] や SURF（Speeded Up Robust Features）[6] などの特徴量を抽出し，それを AdaBoost[7] や SVM（Support Vector Machine）[8] などの機械学習手法を用いて認識するものが一般的であった．しかしながら，画像認識競技会，ImageNet Large Scale Visual Recognition Competition で DNN（Deep Neural Network）が優勝 [9] したことで，時代が変革する．2012 年以降は，物体を認識する手法として DNN が台頭し，そのアーキテクチャの考案 [10, 11]，およびデータセットの大規模化が進んだ．2015 年には，画像から End-to-end で物体検出・認識を同時に行う DNN が登場した．これは，一般物体認識と呼ばれ，YOLO（You Only Look Once）[12] や SSD（Single Shot MultiBox Detector）[13] に代表されるものである．

これらの DNN は，大きな工数をかけ人手により作成されたベンチマークとなるデータセット [14, 15, 16] に対する性能を競い，進化を続けてきた．しかしながら，それらのデータセットで定義されるカテゴリと，ホームサービスロボットが認識すべき日用品・雑貨・家事道具などのカテゴリは構成が異なり，それはロボットが働く家庭・公共空間といった環境によって決定される．そのため，環境に特化したデータセットを作成し，転移学習などの技術を用いて再学習する必要がある．ただし，そのデータセットを毎度人間が作成することは工数の増大を招くため，自動化が望まれる．

また，知的処理の進化により，その計算量は増加傾向にあり，それに対応するためアクセラレータが進化を続けてきた．GPU は，大量の計算コアを搭載し並列計算を実現したデバイスであり，DNN の進化の立役者である．2015 年には GPU に比べ低消費電力・小型なエッジデバイスであるされたモバイル GPU が誕生した．一方で，GPU・モバイル GPU を超える高速計算・低消費電力・小型なエッジデバイスの実現を狙い，FPGA が進化を続けてきた．FPGA とは再構成可能なデジタル回路

であり，アプリケーションに最適化することができるデバイスである．2012 年には，CPU（Central Processing Unit）と FPGA が 1 チップに集積された SoC（System on Chip）である Xilinx Zynq が誕生し，それまでの FPGA 単体と比べ，より柔軟なシステム設計が可能となった．また，同時期に高位合成ツール，Xilinx Vivado HLS（High Level Synthesis）も発表され，それまでの熟練した回路技術者が HDL（Hardware Description Language）を用いて FPGA の内部回路を設計していた状況と比べ，高級言語（C/C++）を用いた設計が可能となった．2015 年には，SoC の CPU と FPGA を包括して設計できる SoC 開発環境，Xilinx SDSoC が誕生した．

しかしながら，これらの技術は確かに進化したが，FPGA の性能を発揮するためには CPU 向けに記述されたアルゴリズムを FPGA 向けに修正する作業が必須であり，それには FPGA の内部回路を熟知した回路技術者を今なお必要とする．ゆえに，ロボット技術者にとっては技術障壁が高く，馴染みが薄いデバイスであることには変わりない．そのため，回路技術者とロボット技術者を繋ぐ統合法の確立が望まれる．

1.2　研究目的

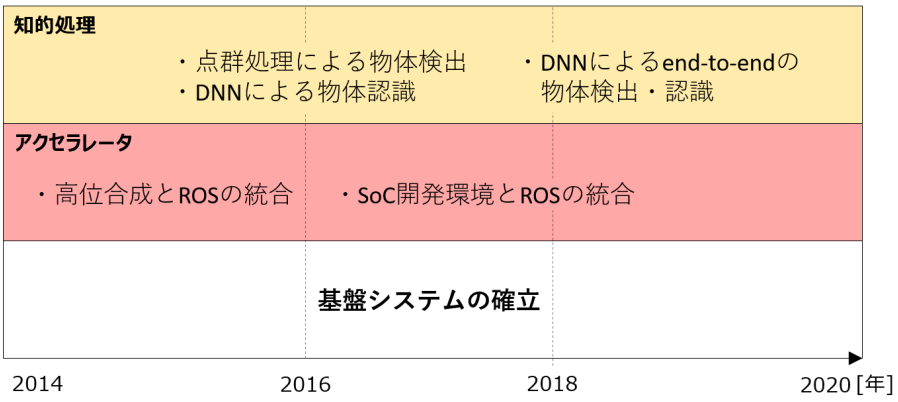


Fig. 1.3　本論文で扱うテーマ

これまで述べた背景をもとに，本研究では Fig. 1.3 に示す，ホームサービスロボットための基盤システムの確立を目指す．また，Fig. 1.3 は横方向が年代を表しており，本研究ではそれぞれの年代に基盤システムと結びつく研究テーマが存在する．これら 4 つのテーマは，大きくホームサービスロボットの知的処理と，そのアクセラレータに大別される．また，それらのテーマにより，ホームサービスロボットの知能の発展と，その高速化が達成される．

点群処理による物体検出・DNN による物体認識では，ホームサービスロボットための DNN による物体認識に向け，データセット作成手法を確立する．また，その DNN をホームサービスロボットに実装し，物体操作の実機評価を行う．DNN による end-to-end の物体検出・認識では，ホームサービスロボットのための DNN による end-to-end の物体検出・認識に向け，データセット半自動作成法を確立する．また，その DNN をホームサービスロボットに実装し，物体操作の実機評価を行う．高位合成と ROS の統合では，ホームサービスロボットの知的処理（特徴量抽出，機械学習）を高位合成を用いて FPGA へとオフロードする．また，高位合成と ROS を統合するインタフェースを確立する．SoC 開発環境と ROS の統合では，ホームサービスロボットの知的処理（DNN）を高位合成を用いて FPGA へとオフロードする．また，SoC 開発環境と ROS を統合するインタフェースの自動生成法を確立する．

1.3　構成

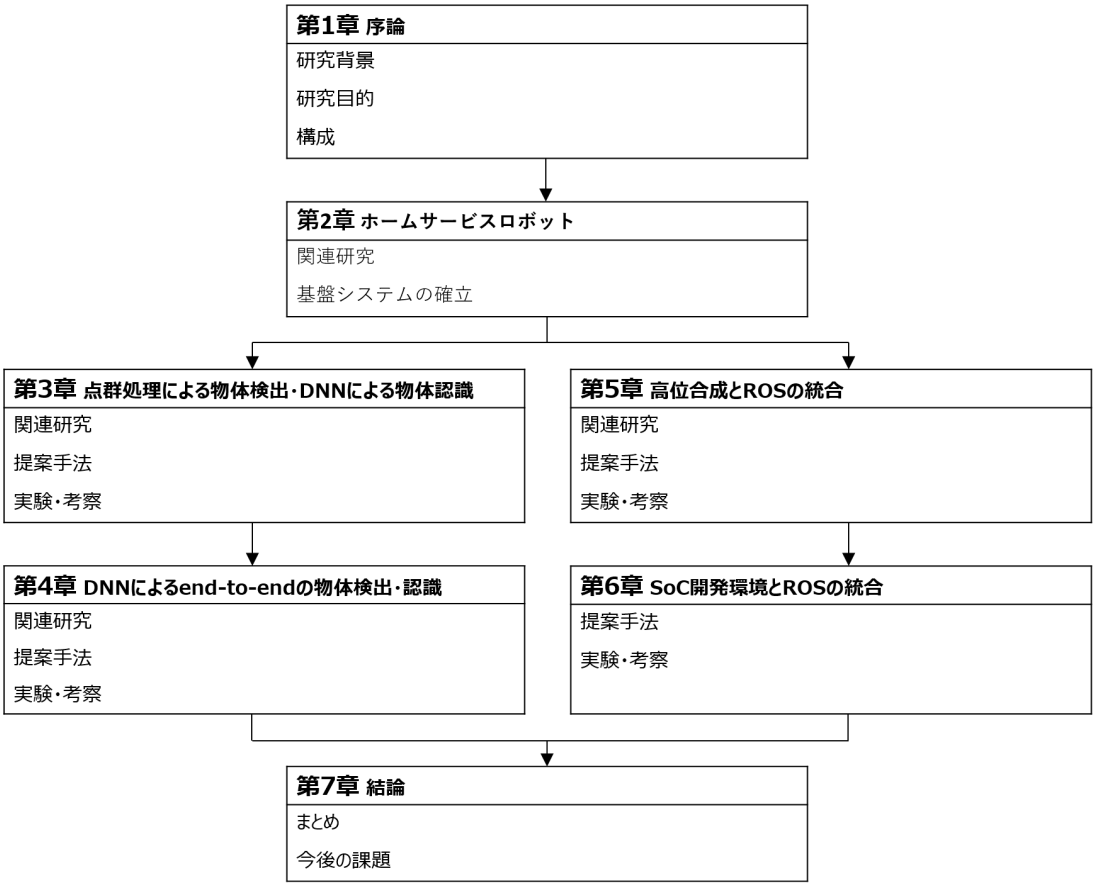


Fig. 1.4　本論文の構成

Fig. 1.4 に，本論文の構成を示す．第 2 章では，ホームサービスロボット全般について述べる．まず，本研究全体に関係する関連研究について述べ，確立を目指す基盤システムについて述べる．第 3 章では，点群処理による物体検出・DNN による物体認識について，関連研究・提案手法を述べ，実験とその結果を述べる．第 4 章では，DNN による end-to-end の物体検出・認識について，関連研究・提案手法を述べ，実験とその結果を述べる．第 5 章では，高位合成と ROS の統合について，関連研究・提案手法を述べ，実験とその結果を述べる．第 6 章では，SoC 開発環境と ROS の統合について，提案手法を述べ，実験とその結果を述べる．第 7 章は，第 3 章から第 6

章の内容をまとめ，今後の課題を述べ，本研究の結論とする．本論文は，第 3 章から第 6 章まで，4 つのテーマを，それぞれ年代に応じた個別の論文として読んでいただきたい．

第 2 章

ホームサービスロボット

2.1 関連研究

2.1.1 ロボットミドルウェア

ここでは，ロボットアプリケーション開発に欠かせない存在となったロボットミドルウェアについて述べる．そもそもミドルウェアとは，OS（Operating System）とアプリケーションの間に入るシステムで，データ通信・管理，デバッグ機能などを提供するものである．Web プログラミング用途など世の中に数多く存在するミドルウェアと，ロボットミドルウェアが決定的に異なる点は，ロボットに搭載されたセンサ・アクチュエータのような計算機以外のデバイスを含めたアプリケーションの開発が可能な点である．現在，ロボット分野においては ROS（Robot Operating System）[2, 17]，OpenRTM-aist[18, 19]，V-Sido OS[20] などのロボットミドルウェアが存在する．今回は，この中でも最も使用人口の多い ROS について解説する．

ROS のあらまし

ROS は米国のロボットベンチャー企業 Willow Garage[21] が，採算性を度外視に，ロボット技術の急速な発展を目指して開発を始めたロボットミドルウェアである．現在は，OSRF（Open Source Robotics Foundation）[22] によって管理が行われている．

ROS が他のロボットミドルウェアに比べ優れていることは，ドキュメント，機能の豊富さである．公式サイト ROS Wiki[17] の年間アクセス数は 117 万アクセス（2019 年）[23] で世界中の情報が集結している．また，ROS に関する学術論文はこれまで

に 5875 本（Google Scholar 2019/08/19 現在）発行されており，最先端の技術が注ぎ込まれている．さらに，これらはオープンソースとして公開されているものがほとんどで，3000 種以上の機能が即時使用可能な状態である．

このような背景から，ROS は世界中の研究機関で採用されており，ロボットミドルウェアのデファクトスタンダードとなりつつある．また企業からも注目を浴びており，商業用途向けのロボットから産業用ロボットに至るまで，今では ROS に対応しているロボットがほとんどである．このことから，「ROS を使用できる = 世界中のロボットを使用できる」といった図式が成り立ちつつある．

ROS のインタフェース

ここでは ROS のインタフェースについて解説する．最初に，以下の解説に必要な ROS 特有の専門用語について述べる．ROS はロボットを機能ごとに細分化した時，その 1 つ 1 つを Node と定義している．そして，その Node を類似したもの同士で分類した単位を Package と定義している．具体的な設計例を上げると，画像処理における 1 つのフィルタ処理が Node であり，画像処理に関わる全機能（例えば複数のフィルタ）を集結したものが Package となる．

この定義のもと，それぞれの Node を 1 つずつプロセスとして起動すると，それらの間でデータの受け渡しを行うプロセス間通信が必要となる．ROS はこれに対し，ネットワーク経由の通信である，Message，Action，Service の 3 種のインタフェースを提供している．

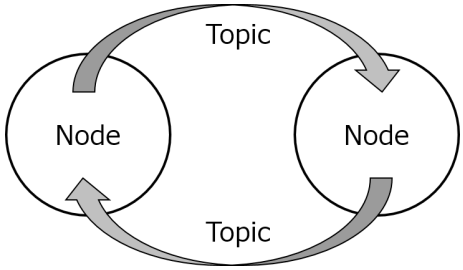


Fig. 2.1 Message のモデル

Fig. 2.1 に 3 種の中でも最も使用頻度の高い Message のモデルを示す．これは出版-購読型モデル（Publish/Subscribe メッセージングモデル）を基にしている．このモデルは，1 対 n の非同期通信，つまり送信者（出版側）が特定の受信者（購読側）を想定せず情報を流すことを可能にしている．具体的には，情報を送信する Node は Message に特定の名前を与えた Topic に対してデータを書き込む．これに対して情報を受信する Node は，Topic からデータを読み込む．これらのデータ操作のタイミングは非同期通信であるため，それぞれの Node が任意の時間で行える．

このような特性から Message を用いることで，複数の Node を繋ぎ合わせて 1 つのアプリケーションを構築できる．また，マルチスレッド処理をユーザが意識せぬまま構築できるため，常にマルチコア CPU（Central Processing Unit）での分散処理に対応した高速なアプリケーションを構築できる．

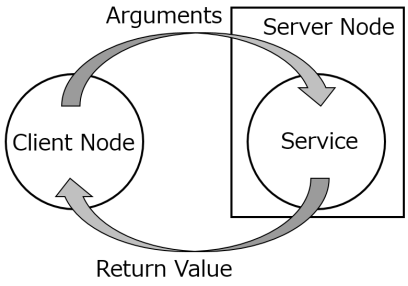


Fig. 2.2 Service のモデル

Fig. 2.2 に Service のモデルを示す．これは前述した Message と異なり，シングルスレッド処理を行うためのモデルである．サーバ側で機能を Service として定義することにより，Client 側からその Service を呼び出すことが可能になる．Service は RPC(Remote Procedure Call) に近い実装である．

次に，Action のモデルを解説するため，Fig. 2.3 を用いてロボットの制御で頻繁に使用される処理のフローを解説する．

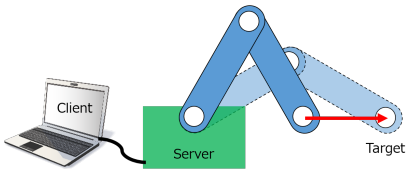


Fig. 2.3　ロボットアームの制御例

これは，ロボットアームの制御を例にしている．まず，クライアントはサーバに対して目標を設定する．次に，サーバが制御を開始する．制御中，サーバは現在の状態（この場合ロボットアームの座標）をクライアントに逐次的にフィードバックする．無事目標まで到着すると，サーバはクライアントに対して終了信号を出力する．

この例において，クライアントは目標を，サーバはフィードバックと終了信号をそれぞれ出力しており，これに合わせたインタフェースが Action である．

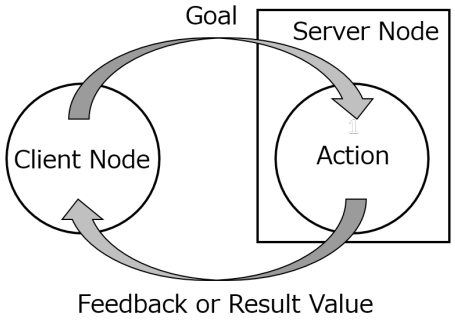


Fig. 2.4　Action のモデル

Fig. 2.4 に Action のモデルを示す．Action は Message と比べ，サーバから終了信号が出力されるまでクライアントが待つことが特徴的である．Message を用いて同じ振る舞いをするためには，2 つの Topic が必要でコーディングが効率的ではない．またフィードバックが存在するため，Service に比べ長期間の終了待ちを要する処理に用いられる傾向がある．

ROS によるロボットアプリケーション開発

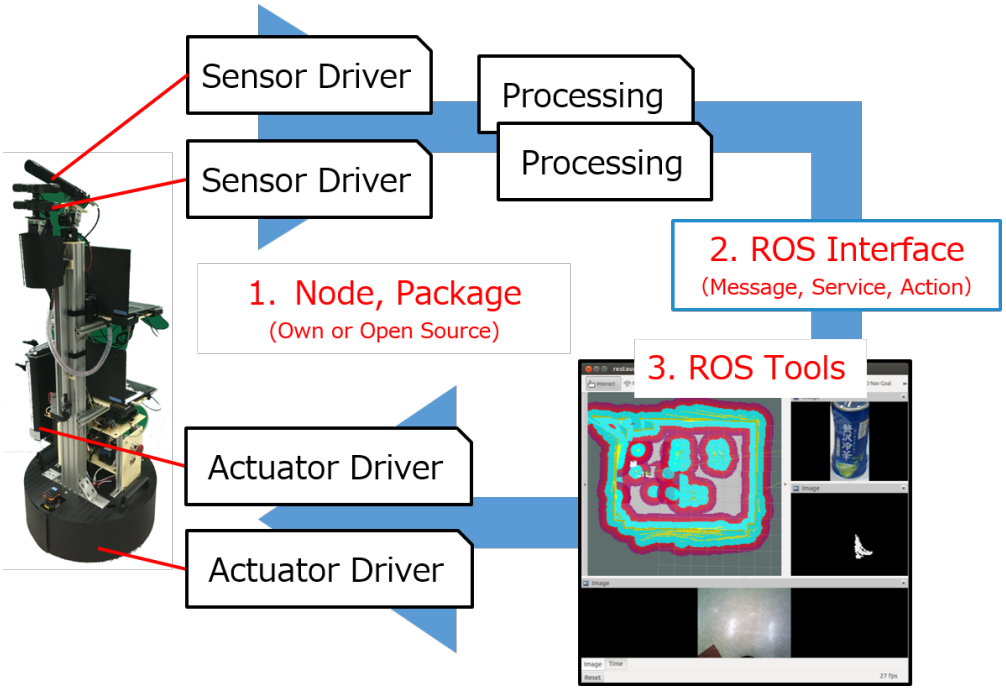


Fig. 2.5　ROS によるロボットアプリケーション開発

ROS によるロボットアプリケーション開発について Fig. 2.5 にまとめる．ROS によるロボットアプリケーション開発では，最初の段階として Node，Package の設計を行い，個々の機能を構築する．これらは，自ら設計を行う手法，オープンソースで公開されているものを用いる手法，どちらの手法も取れる．

次の段階として，個々の機能は前述した ROS のインタフェースによって繋ぎ合わされる．これにより，個々の機能は統合された状態となり，ロボットの振る舞いとなる．また統一されたインタフェースにより繋ぎ合わされることで，後々の機能の入れ替え，ロボットの振る舞いの変更が容易である．

これまでの 2 段階を通してロボットが動き出すと，開発はデバッグ段階へと移る．この時，ロボットの関節角やセンシング情報などの可視化が重要となる．これに対して ROS は用途に応じた GUI ベースのデバッグツールを提供しており，ロボットアプリケーションを円滑に完成へと導ける．

2.1.2　構成と特性

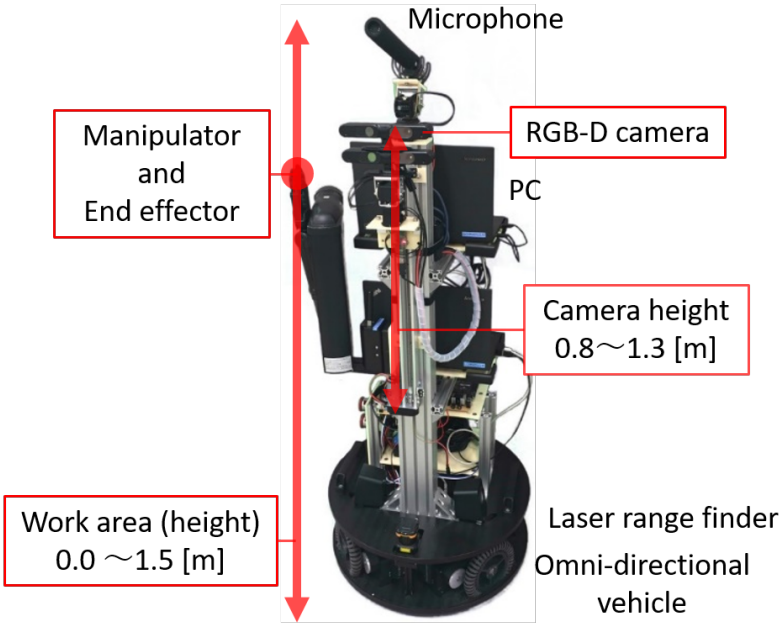


Fig. 2.6　ホームサービスロボット, “Exi@”.

Fig. 2.6 に，筆者らが開発するホームサービスロボット “Exi@”[24] の構成を示す．ホームサービスロボットは，家庭・公共空間で人間と共存しながら働くことが求められている．すなわち，人間が暮らす環境で，日用品・雑貨・家事道具などを操作し働く必要がある．そのため，ロボットは人間の五感・運動機能に近い構成となっている．具体的には，耳に相当するマイク，目に相当する RGB-D カメラ・LRF (Laser Range Finder)，腕に相当するマニピュレータ，脚に相当する移動台車が備わっている．ホームサービスロボットの特性として，環境に合わせて多様な視点をとり作業を行うことが挙げられる．ロボットは，床上や家具内の物体を操作するため，エンドエフェクタが Fig. 2.6 に示すよう，床面から高さ 0.0〜1.5m の範囲に到達する．また，エンドエフェクタの稼働範囲に合わせ広い視野を確保するため，カメラがパンチルト，および Fig. 2.6 に示すよう床面から高さ 0.8〜1.3m の範囲で上下動する．これらを生かし，ロボットは具体的に，床を見下ろしたり，高さ 1.0m 以上の棚を真横から見たりし，作業を行う．他の特性として，物体を操作する時に正確な 3 次

元座標を必要とするため，カラー画像以外に赤外線による測距・3 次元計測機能が加わった RGB-D カメラが搭載されていることが挙げられる．また，他の Exi@以外のロボットも同様のハードウェア構成となっており，Totota HSR[25]，Fetch mobile manipulator[26]，PAL Robotics Tiago[27] などが存在する．

2.1.3　ベンチマークテスト

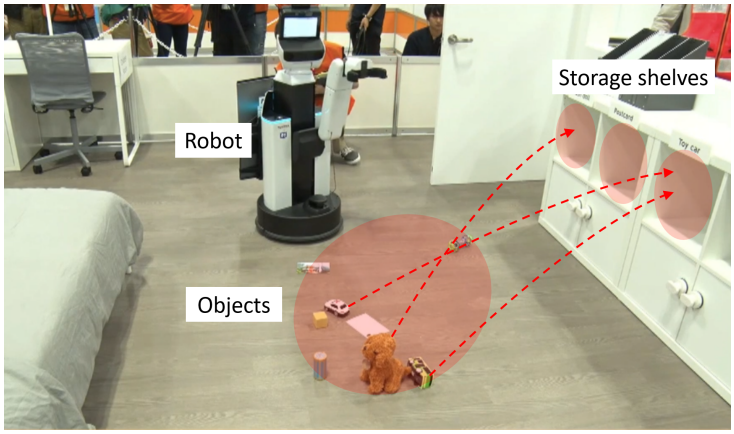


Fig. 2.7　WRS のアリーナ

ホームサービスロボットのベンチマークテストとして，RoboCup@Home[28, 29] や WRC[30] が開催されている．ここでは特に，本研究で評価に用いた WRC の競技 Tidy Up Stage 1 について述べる．WRC は世界規模のロボット競技会であり，家庭内で様々な補助を行うホームサービスロボットの実現を目指している．そして，Tidy Up Stage 1 のルール [31] は，まずロボットが Fig. 2.7 に示す競技アリーナ（子供部屋）へと自律移動し入場する．次に，Fig. 2.8 に示す全 15 個のおもちゃのうち 10 個が床に散らかっており，ロボットはそれを探しマニピュレータで把持し収納場所の棚まで持ち運ぶ．この時，Fig. 2.7 に示すように収納場所は複数存在し，Fig. 2.7 の矢印のようにおもちゃのカテゴリに応じて振り分けて収納しなければならない．おもちゃの振り分け方は競技開始前に予め定義されており，また収納場所の中に入れることができれば収納する位置・方向は問われない．おもちゃを正しい収納場所に入れた場合は 5 点，間違った収納場所に入れた場合は 3 点が得られる．おもちゃを如何なる収納場所にも入れられない場合，得点できない．競技時間は 12 分で，得点の高さで順位が決められる．

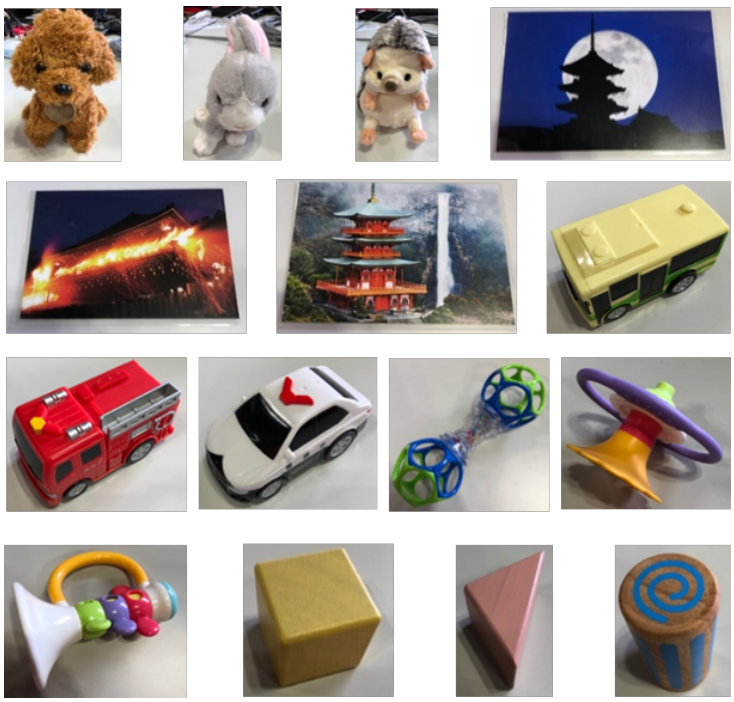


Fig. 2.8　WRS のオブジェクト

このルールで得点するためには，まずロボットはおもちゃの位置を捉え，マニピュレータの制御目標を決定しなければならない．次に，おもちゃのカテゴリを認識し，正しい収納場所を判断しなければならない．すなわち，物体検出・認識が出来なければ収納場所までおもちゃを運べず，その性能が得点に大きく関わる．また，Tidy Up Stage 1 では標準環境が設定されていない．家庭を模した競技アリーナということのみが定義されており，ロボットはセットアップ日になるまで具体的な環境を知らない．また，物体（おもちゃ）に関しても同様である．そのため，将来ホームサービスロボットが働く環境と同じく，ロボットが未知の環境で物体を検出・認識できるよう，迅速な DNN の訓練および，その仕組みが必要となる．

2.1.4 CPU，GPU，FPGA の比較

近年，CPU に代わる計算機が注目を浴びている．これは，技術発展により増大しつづける計算負荷に対して，高速計算を可能にするためである．このような計算機として，GPU，FPGA が挙げられる．GPU は大量のコアを搭載し，並列計算を実現する．FPGA は再構成可能なディジタル回路であり，並列性と柔軟性を併せ持つ．Table 2.1 に CPU，GPU，FPGA の比較結果をまとめる [32].

Table 2.1 CPU，GPU，FPGA の比較結果

Evaluation Item	CPU	GPU	FPGA
Speed	× 1	× 14	× 228
Performance Per Watt	× 1	× 31	× 584
Development Days	× 1	× 45	× 300

これは，Smith-Waterman アルゴリズムをベンチマークとして比較した結果である．この結果より，FPGA は計算速度で，CPU の 228 倍，GPU の 16 倍もの性能を有していることが分かる．また，単位電力あたりの計算性能でも，CPU の 584 倍，GPU の 17 倍もの性能を有していることが分かる．しかしながら，開発工数は CPU の 300 倍，GPU の 7 倍必要な短所も見受けられる．これは，FPGA の持つ高い計算性能を引き出すため，専用アルゴリズムを開発する必要があるためである．また，ディジタル回路であるため，実装，デバッグにも工数を要す．

これらを踏まえ，GPU は比較的手軽に計算を高速化できる計算機だと考えられる．また，FPGA は完成までに時間を要すが，得られる計算性能・低消費電力さには目を見張るものがある．

2.1.5 ASIC，FPGA の比較

ASIC（Application Specific Integrated Circuit）は特定アプリケーションをターゲットとした IC チップである．Fig. 2.9 に ASIC と FPGA を比較した結果を示す．

ASIC は実装時最適化されているため，計算性能，消費電力の低減といった面では FPGA に比べ有利である．また，同じ IC を大量に使用するアプリケーションの場

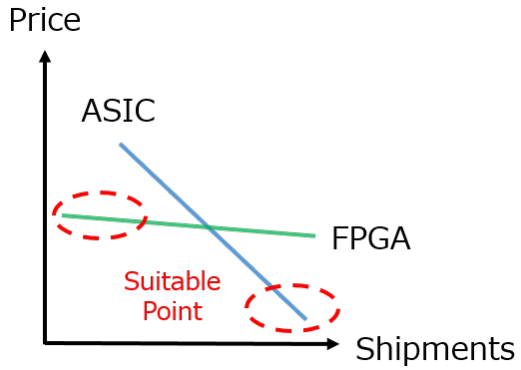


Fig. 2.9　ASIC，FPGA の比較

合，大量製造によるコスト削減が可能となり，FPGA に比べコスト面でも有利である．これに対し FPGA は，再構成可能という特徴を生かし，何回でも機能の書き換えが可能である．これにより，後々アップデートを行うようなアプリケーション，また大量製造を行わないアプリケーションにおいて ASIC に比べ有利である．

2.1.6　ロボットに最適なアクセラレータ

ここまで，GPU，ASIC，FPGA といったアクセラレータを解説した．ここで，ロボットのような組み込み環境に最適なアクセラレータについて検討する．

まず，ホームサービスロボットや自動運転車においては，前述したような知的処理の効率化が求められている．GPU は知的処理の実装が比較的容易であり，CPU に比べある程度の高速化も可能である．GPU は，ラピッドプロトライピングとしての実装に適している．しかしながら，組み込み環境では，消費電力が高いこと，それに伴う排熱が問題となる．

次に ASIC は，頻繁に使用する特定処理の効率化には絶大な効果を示す．しかしながら，知的処理アルゴリズムは日進月歩で進化しており，後々機能を書き換えられない ASIC の場合，陳腐化が懸念される．

最後に，FPGA は消費電力，排熱，機能書き換え全ての面において問題はないと考えられる．ただし，開発工数の多さが問題として残る．この理由として，知的処理の実装だけではなく，インタフェースの開発，デバッグ作業に工数を多く必要とすることが挙げられる．ゆえに，ロボットと FPGA を繋ぎ合わせる高い親和性を有したイ

ンタフェースを構築することで，FPGA がアクセラレータとして最も適すると考えられる．

2.1.7 システムへの FPGA の統合

hw/sw 複合体

従来のアクセラレータを含めたシステムは，CPU と GPU で構成されるものがほとんどであった．これに対し，CPU と FPGA で構成した hw/sw（ハードウェア/ソフトウェア）複合体 [33] と呼ばれるシステムがある．このシステムでは，これまで CPU（ソフトウェア）で行っていた処理の一部を FPGA（ハードウェア）にオフロードすることで，システム全体で高速計算，消費電力の低減を目指している．この時，オフロードする処理としては，信号処理など FPGA が得意とするものが挙げられる．また，複雑な条件分岐など FPGA が苦手とするものは従来通り CPU にて処理を行う．これにより，CPU と FPGA の長所を兼ね備えたシステム構築が可能となる．

hw/sw 複合体を用いた知的処理の効率化

Hsiao らは，hw/sw 複合体を用いた人物検出の効率化を提案している [34]．この研究では，前処理，HOG 特徴量抽出，SVM による人物・非人物の認識というアルゴリズムを効率化の対象としている．このアルゴリズムの計算負荷を分析すると，HOG 特徴量抽出の負荷が支配的となる．Hsiao らは，この HOG 特徴量抽出を FPGA にオフロードし，CPU と FPGA を繋ぐインタフェースを構築することにより効率化に成功している．

具体的な結果として，FPGA（Xilinx XC6SLX150T）により PC（Intel Core i7-3770/3.4GHz DDR3/8GB）と比べ HOG 特徴量抽出が 20 倍高速化されている．また，組み込み CPU（Colibri T20Colibri T20）と FPGA で構成したシステムにより，15fps での人物検出を可能としてる．

この結果をまとめると，hw/sw 複合体は知的処理の効率化に対して有効な性能を有している．また，この研究では CPU と FPGA を繋ぐインタフェースを 1 から構築している．ゆえに，このインタフェースを自動生成すると開発工数の削減に繋がる．

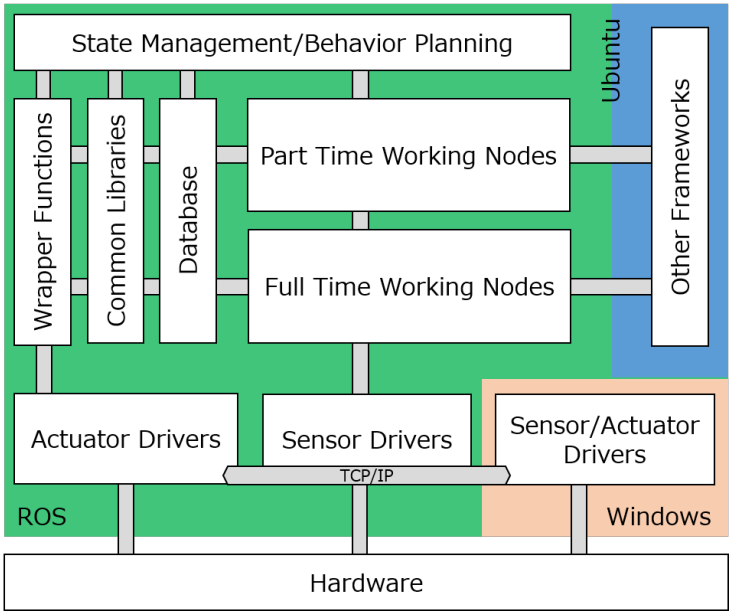


Fig. 2.10　ホームサービスロボットのシステム図

2.2　基盤システム

Fig. 2.10 に，筆者らが開発するホームサービスロボットのシステム図を示す．このシステムはソフトウェアを，状態管理・動作計画，一時的 Node，常時的 Node，ドライバ類，フレームワークに分けて管理している．これは，増大傾向にあるソフトウェアをロボット上の PC で使用するための工夫である．状態管理・動作計画部は，現在の状態に応じて，一時的 Node を起動したり終了したりする．具体的には，物体を扱う時には，物体検出・認識ノードのみを起動し，人間とインタラクションする時には人物検出・認識ノードのみを起動する．これにより，必要ない Node を停止することが可能となり，計算資源の節約に繋がる．そして計算資源に余裕を持たせることでシステム全体の安定動作に繋がる．

次に，Fig. 2.11 にソフトウェアブロック図を示す．大きく，音声インタラクション，画像処理，マニピュレータ制御，自律移動のブロックから成り立っている．本研究では，特に人物検出・追跡，物体検出・認識を行う画像処理に着目し，その知的処理とアクセラレータを提案する．

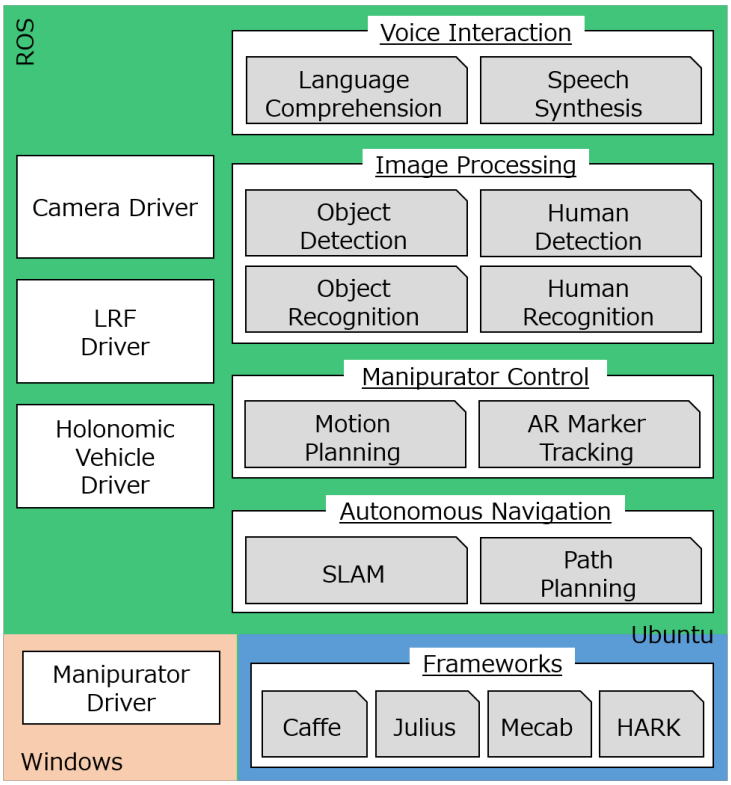


Fig. 2.11 ホームサービスロボットのソフトウェアブロック図

第 3 章

点群処理による物体検出・DNN による物体認識

本テーマでは，ロボットアームにより正確に物体を把持するための，3 次元点群処理による物体検出，DNN を用いた頑健な物体認識，マーカーベースによるロボットアームのビジュアルフィードバック制御を組み合わせたホームロボット向け物体認識・把持システムの構築を述べる．

3.1 関連研究

筆者らの研究グループでは，ロボットアームのビジュアルフィードバック制御に関する研究 [35] を行ってきた．この研究では，RGB-D カメラを用いた色空間ベースの画像処理システムによりエンドエフェクタ座標を算出しフィードバック制御することで，位置決め精度が低いロボットアームの性能を補償するシステムを提案した．実験により，ビジュアルフィードバック制御がない場合の物体把持成功率 48% に比べ，提案システムを用いることで 72% まで成功率が向上する結果が得られた．

しかし，把持対象物体の認識には，OpenCV の SURF 特徴量 [36] と，確率的簡易マッチング手法 [35] を採用しており，認識率に問題があった．また，エンドエフェクタ座標の算出に色空間ベースの画像処理を用いていたため，物体や背景の色によっては正しく把持できない問題点が挙げられた．

3.2　提案手法

本研究では，Fig. 3.1 に示す一般的な家に存在する机の上に置かれた物体を，認識・把持することを目指したシステムを提案する．ハードウェアの改造として，マーカーベースのビジュアルフィードバックを可能にするため，ロボットアームのエンドエフェクタに，Fig. 3.2 の左側に示すようなマーカーを貼った．提案システムの処理フローを以下に説明する．

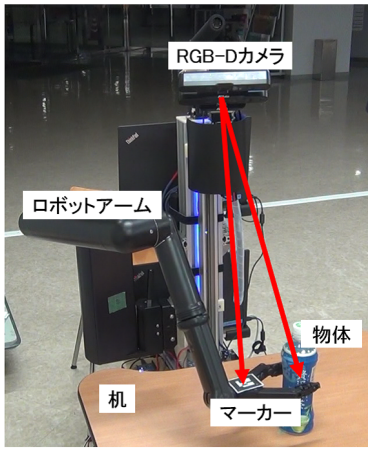


Fig. 3.1　提案システムによる物体把持

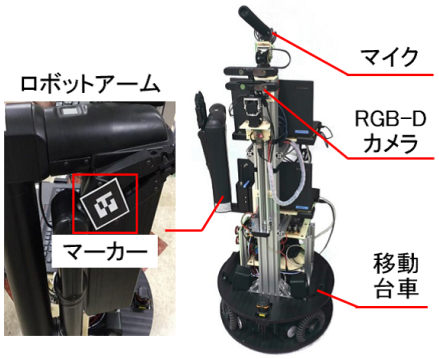


Fig. 3.2　ホームロボットのハードウェア

(1) 注目 3 次元点群を生成

Fig. 3.3(a) に示す RGB-D カメラから得られる 3 次元点群に対して，PCL（Point Cloud Library）を用いて指定範囲で点群を切り出すパススルーフィルタ [37] をかけることにより，Fig. 3.3(b) に示す注目 3 次元点群を生成する．

(2) 机の天板を検出

注目 3 次元点群に対して PCL の RANSAC 法 [37] を用いて平面検出を行い，Fig. 3.3(c) に青く示す机の天板を検出する．

(3) 物体の 3 次元点群の生成

Fig. 3.3(d) に示すように，検出された机の天板上に置かれた物体の 3 次元点群を切り出す．

(4) 物体画像の生成

物体の 3 次元点群をもとに，RGB 画像から物体画像を切り出す．

(5) 物体認識

切り出された物体画像を DNN により認識する．

(6) 物体座標の算出

物体の 3 次元点群をもとに，Fig. 3.4 に示す RGB-D カメラ座標系から見た物体の重心座標を算出する．

(7) 座標変換

物体の重心座標を，Fig. 3.4 に示すロボット座標系に座標変換し，目標座標とする．

(8) ビジュアルフィードバック

Fig. 3.1 に示すように，ar_track_alvar[38] を用いて RGB-D カメラによりエンドエフェクタのマーカーを検出し，目標座標に対して誤差を最小化するフィードバック制御を行う．

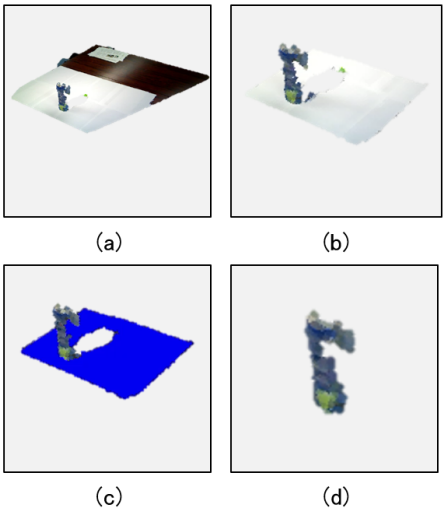


Fig. 3.3　3 次元点群処理

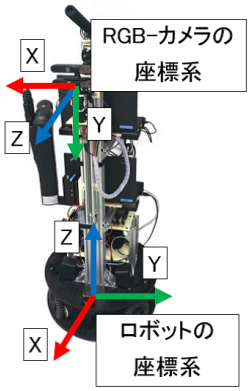


Fig. 3.4　RGB-D カメラとロボットの座標系

3.3　実験・考察

3.3.1　物体認識システムの構築

Caffe[39] と呼ばれる Deep Neural Networks のフレームワークと OpenCV を用いて，以下の 3 方式の物体認識システムを構築した．

(A) Shallow CNN

Fig. 3.5 に示す，畳みこみ層とプーリング層が 3 層，全結合層が 3 層（MLP，

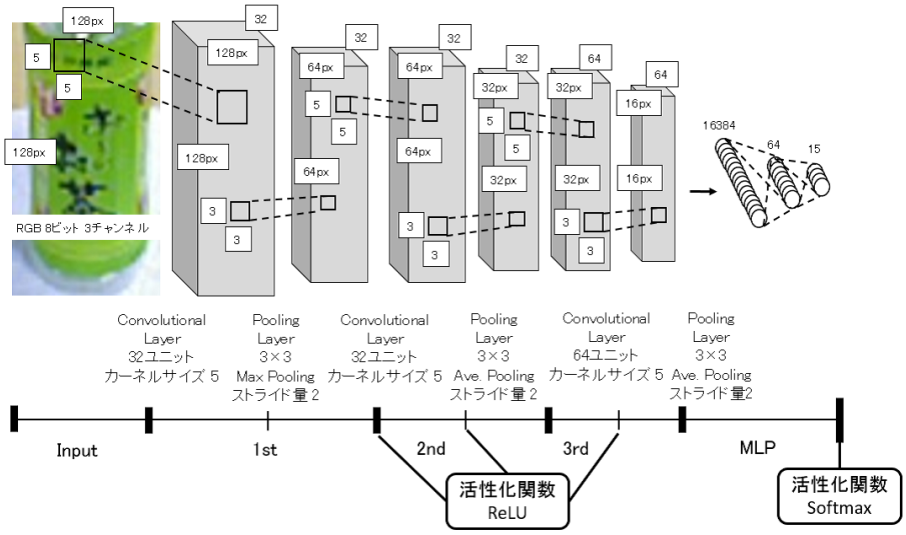


Fig. 3.5 実験に用いたネットワーク構造

Table 3.1 物体形状によるクラス分け

クラス	物体番号	形状
Class1	1, 2, 3	ペットボトル
Class2	4, 5	紙パック
Class3	6, 8, 9, 10, 15	円筒
Class4	7	直方体
Class5	11, 12, 13, 14	その他

Multi-Layer Perceptron）の CNN（Convolutional Neural Networks）を用いて物体認識を行う．構造が浅く，入力から出力まで時間を要さないことが特徴である．入力画像には，物体画像を OpenCV のバイリニア補間により 128px × 128px，RGB 3 チャンネル 各 8 ビットにリサイズしたものを用いた．

(B) HOG + SVM + Shallow CNN × 4

（A）を改良したものである．Fig. 3.6 に示す学習対象の物体は，Table 3.1 に示すように，ペットボトル，紙パック，円筒，直方体，その他の 5 つの形状に分けられる．そこで，最初に OpenCV の HOG 特徴量と SVM[36] を用いて形状によって 5 クラスに分ける．次に Class1〜Class3，Class5 それぞれを

Fig. 3.6　学習した物体

（A）と同じニューラルネットワークで学習し，認識を行う．これにより，1 つのニューラルネットワークで取り扱うクラス数を減らし，認識性能の向上を狙っている．

(C) GoogLeNet[40]

ImageNet Large Scale Visual Recognition Challenge 2012 [41] のデータセットにより学習済の 22 層の CNN を再利用 [42] したものである．層数が多く，全体の学習には膨大な時間を要するため，最終層のみを再学習する転移学習により今回の問題への適用を行った．入力画像には，物体画像を OpenCV のバイリニア補間により 224px × 224px，RGB 3 チャンネル 各 8 ビットにリサイズしたものを用いた．

3.3.2 物体認識の評価

構築した物体認識システムの性能を評価した．Fig. 3.6 に今回学習した 15 種類の物体を示す．これらは，2016 年 3 月に行われたホームロボットの国際競技会 RoboCup@Home ジャパンオープンで使用された物体である．1〜10，13〜15 のように個体差がない物体と，11，12 の果物のような個体差がある物体が用意されていた．この物体を RoboCup の会場で，Fig. 3.7 に示すように回転させ様々な角度から 100 枚撮影した．これに画像処理を行い，RGB それぞれのチャンネルを 0.9 倍，1.0 倍，1.1 倍にした全組み合わせ 27 パターンのノイズを加え，1 物体に対して 2700 枚のデータセットを構築し，そのうち 2000 枚を学習用画像とした．学習には，Core i5-3470，DDR3 8GB，GTX970 の PC を用いた．

検証は，RoboCup の会場と異なる場所で行った．すなわち，各物体を正しく認識できるか検証するとともに，撮影場所と検証場所を変えることで，光源など環境変化に対して頑健に認識が可能かを検証した．検証には同一物体が用意できた 1〜10，類似物体が用意できた 11，12 を用い，30 度ずつ 12 方向から物体画像を生成して評価した．検証には Core i5-5200U，DDR3 12GB の PC を用いた．

実験結果を Table 3.2 に示す．（A）は入力から出力まで 0.2s と処理速度が高速ではあるが，正解率が 59% と頑健な認識は不可能であった．また（A) を改良した（B) は，3% の正解率向上にとどまった．（C）は正解率 99% と頑健な認識結果を示した．また，類似物体である 11，12 に対しても実用に耐える正解率を示した．しかしながら，(C) は 22 層の CNN とネットワークが大きいため，処理に 4.6s もの時間を要した．

これらの結果より，（A）と（C）の処理速度の差 4.4s は，現状のロボットの全行動時間を考慮すると大きな問題にはならず，頑健な認識が可能な（C）の方式で物体認識を行うことにした．今後，カメラに映る物体数が多い状況下で動作させるには，高速化を検討しなければならない．

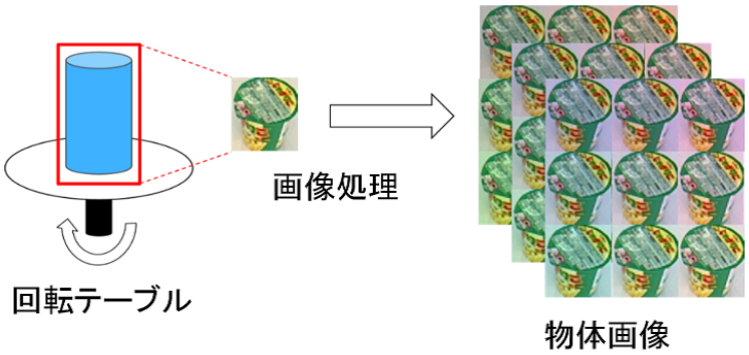


Fig. 3.7　データセットの生成

Table 3.2　各ネットワークにおける正解率

方式		物体番号												正解率 [%]	処理速度 [s]
		1	2	3	4	5	6	7	8	9	10	11	12		
(A) Shallow CNN	正解数	8	10	1	10	0	0	10	12	3	12	11	8	59	0.2
	誤認数	4	2	11	2	12	12	2	0	9	0	1	4		
(B) HOG + SVM + Shallow CNN × 4	正解数	11	2	4	10	11	10	11	10	6	9	4	1	62	0.3
	誤認数	1	10	8	2	1	2	1	2	6	3	8	11		
(C) GoogLeNet2014	正解数	12	12	12	11	12	12	12	12	12	12	12	12	99	4.6
	誤認数	0	0	0	1	0	0	0	0	0	0	0	0		

3.3.3　ビジュアルフィードバック制御の評価

3 次元点群処理による物体座標の算出と，それに対するマーカーベースのビジュアルフィードバック制御を行ったときの，物体把持成功率を求める実験を行った．この実験では Fig. 3.1 に示す環境で，机上の物体に対して 30 回，物体把持を行った．前述したように先行研究 [35] の成功率は 72% であるため，本実験ではそれ以上の成功率を目標とした．

ビジュアルフィードバック制御がない場合，先行研究，提案手法の成功率を Table 3.3 に示す．結果として，提案手法は 28 回物体把持に成功し，93% の成功率が得られた．ビジュアルフィードバック制御がない場合に比べ 45 ポイント，先行研究に比べ 21 ポイントの成功率向上が確認された．これは，エンドエフェクタの検出をマーカーベースにすることにより，物体や背景の影響を受けなくなったためである．

なお，失敗した 2 回はどちらも，物体座標が誤って算出されたことが原因であった．この対策として，ビジュアルフィードバック制御中に目標座標を逐一更新しつづけるようシステムを変更すべきである．

Table 3.3 先行研究との比較

	ビジュアルフィードバック制御なし	先行研究 [35]	提案手法
成功率	48%	72%	93%

第 4 章

DNN による end-to-end の物体検出・認識

本章では，前章で述べた物体検出・認識の高速化を狙い，ホームサービスロボットへ一般物体認識を応用する．また，その時のデータセット半自動生成法について述べる．

4.1 関連研究

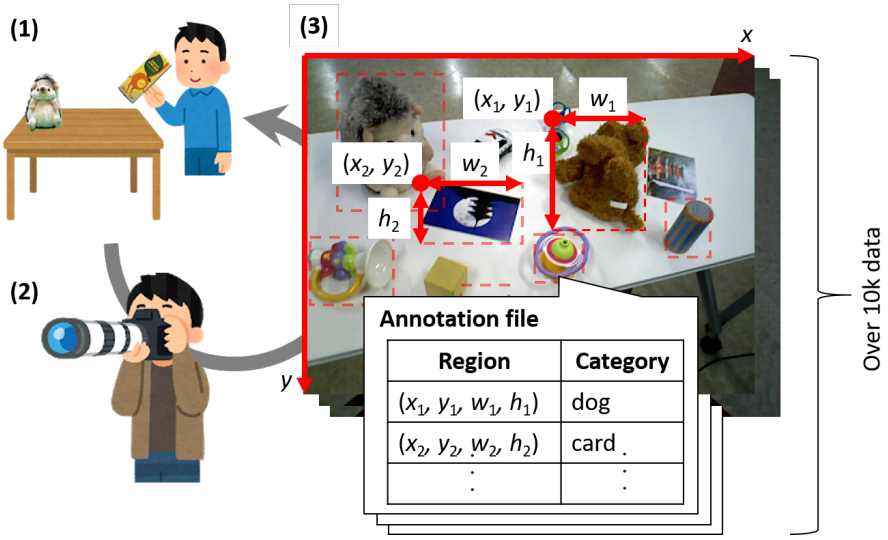


Fig. 4.1 人手によるデータセット作成の手順

本節では，これまでに報告されているデータセットを作成する手法について概要と課題を述べる．まず，人手によりデータセットを作成する手法があり，Fig. 4.1 に示す手順で作成される．以下に Fig. 4.1 内の番号と照らし合わせて解説する．

(1) 環境内の床・家具などの場所をランダムに選び，物体をランダムな数選び，ランダムな位置，方向で置く．
(2) ランダムなカメラアングルを決定し，撮影する．
(3) 撮影した画像に対して，物体の領域（基準座標 (x, y) と横・縦の大きさ (w, h)）・カテゴリを示した，DNN に教師信号として与えるアノテーションファイルを記述する．

そして，（1）～（3）を数万回繰り返しビックデータ化する [43]．しかしながら，この作業は手順および考慮・選択すべきパラメタ（場所，物体の種類・数・位置・方向，撮影アングル）が多く，また日常生活で日々物体が増減するため，各環境に対して人手により継続して行うことは現実的ではない．なお，Fig. 4.1（1）～（3）を 1 サイクルこなすには数分の時間を要し，それを数万回繰り返すと，1 週間から 1 カ月程度データセット作成に時間を要する．

次に，Georgakis らの手法 [44] を述べる．本手法は，画像合成技術により，データセット作成をシステム化し，人間の作業負担を軽減する．本手法では複数の条件でデータセットを作成し，それらを用いて DNN を訓練し，それぞれ物体検出・認識の性能指標である mAP（mean Average Precision）で評価している．それらの中で最も高い値が得られた条件（SP-BL-SS，Selective Positioning-Blending-Selective Scale）の具体的な処理の流れを以下に述べる．

(1) RGB・Depth 画像が提供される GMU Kitchen Scenes データセット [45]，Washington RGB-D Scenes v2 データセット [46] を背景とし，セグメンテーション [47] および平面検出 [48] を用いて，物体が置かれる床と平行な平面を抽出する．
(2) RGB・Depth 画像が提供される BigBIRD[49]，Washington RGB-D v1 データセット [50] から，グラフカット [51] を用いて，物体のみを抽出した画像を作成する．
(3) 抽出した平面上で物体を合成する点を決定し，その点の距離情報を背景の Depth 画像から取得し，距離に応じて物体が正しい大きさになるようスケーリ

ングする．

(4) 背景に対して物体を，Fast Seamless Cloning[52] を用いて合成する．

(5) 物体を合成した領域・カテゴリは既知であるため，同時にアノテーションファイルを自動的に作成する．

本手法で作成したデータセット（合成データ）で，SSD を訓練した場合，mAP が 33.5 となった．また，人手で作成したデータセット（実データ）で訓練した場合，mAP は 65.6 となった．また，合成データ 100% に実データを 10% 追加し訓練した場合，mAP は 71.6 となり，実データのみの場合よりも値が向上した．本手法の課題として，背景・物体ともにデータセットの画像を使用しており，その撮影法については議論されていないことが挙げられる．そのため，背景・物体ともにロボットの視点を考慮しておらず，特に背景についてはロボットが必ず平面（物体）を見下ろす視点であることが仮定されている．また，合成データで訓練した DNN をロボットに実装した評価を行っておらず，ホームサービスロボットに応用できるか議論されていないことも挙げられる．

本研究で我々は，Georgakis らの手法と同様に，データセット作成をシステム化し，人間の作業負担を軽減する．また，2.1.1 節で述べた，ホームサービスロボットの多様な視点に対応するため，Georgakis らの手法と比べ，背景に対し物体を合成する点の教師方法を変更する．これにより，ホームサービスロボットへの応用を考慮したデータセット作成法を提案し，実機評価まで行う．

4.2 提案手法

本研究では，ホームサービスロボットの物体検出・認識のためのデータセット半自動作成法を提案する．本手法では予め，ロボットが物体を検出・認識する時にとる姿勢に関するパラメタとして，RGB-D カメラと物体との間の距離 d，RGB-D カメラの床からの高さ h およびチルト角 θ，それぞれの範囲 d_{min}，d_{max}，h_{min}，h_{max}，θ_{min}，θ_{max} を決定する必要がある．提案手法の概要を Fig. 4.2 に示し，以下にデータセット作成手順を Fig. 4.2 内の番号と照らし合わせて述べる．

(1) まず，RGB-D カメラを用いて人手により環境内の様々な場所を背景画像（RGB・Depth 画像）として撮影する．この時，RGB-D カメラと場所との間の距離が d_{min} から d_{max}，RGB-D カメラの床からの高さが h_{min} から h_{max}，

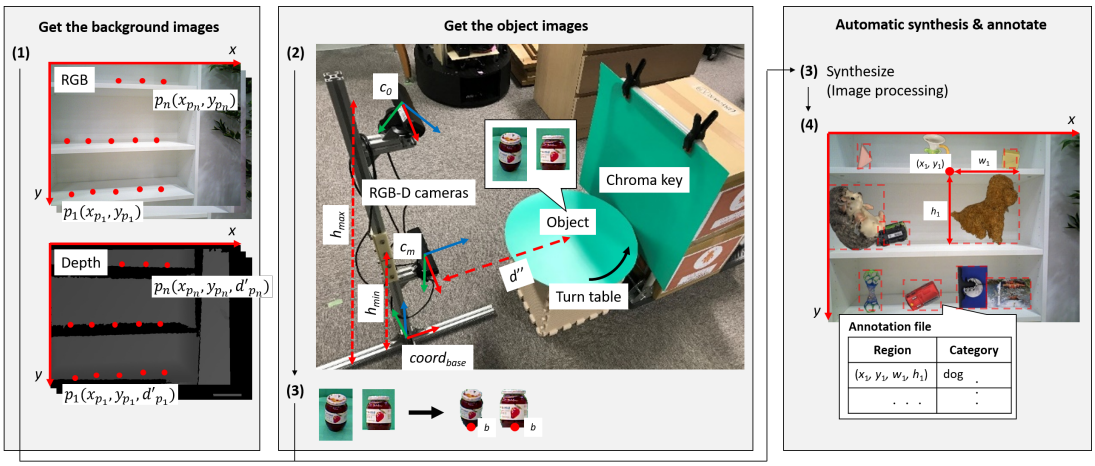


Fig. 4.2 ホームサービスロボットの物体検出・認識のためのデータセット半自動生成法

およびチルト角が θ_{min} から θ_{max} の範囲内で，RGB-D カメラを設置する．次に，撮影した背景画像に対し，物体が置かれるであろう点 p を設定する．設定には OpenCV を用いて独自実装した GUI（Graphical User Interface）を用い，RGB 画像上でクリックした座標を点 $p(x_p, y_p)$ として設定する．また，Depth 画像で点 $p(x_p, y_p)$ を参照し，距離 d'_p を求める．Depth 画像から距離が得られない場合は，キーボード入力で距離を指定する．また，RGB 画像は，色恒常性仮説の一つである灰色仮説 [53] に基づき補正する．

(2) RGB-D カメラ，ターンテーブル，均質な色の背景（Chroma key）で構成された装置を用い，各物体を撮影する．この時，m 個のカメラ，c_0〜c_m を用いる．カメラ c_0 は床からの高さ h_{max} およびチルト角 θ_{max} で設置する．カメラ c_m は床からの高さ h_{min} およびチルト角 θ_{min} で設置する．その他のカメラは，床からの高さ h_{min} から h_{max} の間，およびチルト角 θ_{min} から θ_{max} の間で設置する．また，カメラ c_m と物体の間の距離を d'' と定義し，全てのカメラはカメラ c_m の垂直方向の延長線上に設置する．ターンテーブルは物体を回転し，様々な方向から物体を撮影することに貢献する．均質な色の背景はクロマキーとして用いられ，以降で画像処理を行うとき，物体とそれ以外の判別を容易にする．物体は点群として撮影し，全て同じ座標系で共通の処理を行うため，それぞれのカメラ座標系から $coord_{base}$ 座標系へと変換する．変換された点群に対し，PCL（Point Cloud Library）により，ターンテーブルの平面を抽出 [48]

する．点群からターンテーブルより外側およびターンテーブルを除去することで，物体の点群を得る．そして，点群中の物体領域を，RGB 画像中の物体領域へと PCL を用いて変換することで，物体が切り出された RGB 画像を得る．また，物体が切り出された RGB 画像は，灰色仮説に基づき補正する．

(3) 物体が切り出された RGB 画像に対して，HSV 変換を行い，閾値処理をして均質な色の背景を抽出し，透過する．また，以降の処理で用いるため，物体画像の底部中央を点 b とする．

(4) 背景画像と物体画像を画像処理により合成する．最初に，背景画像をランダムに 1 枚選択し，物体画像をランダムに l 枚選択する．また，物体画像は θ'_{min} から θ'_{max} の範囲でアフィン変換を行い回転する．そして，それぞれの物体画像に対して，背景画像の点 p をランダムに割り当てる．割り当てられた点 p の距離に対して，物体画像が正しい大きさとなるよう，式 4.1 を用いて係数 k を求め伸縮処理を行う．最後にそれぞれの物体画像の点 b が，背景画像の点 p と一致するように合成する．

$$k = d''/d'_p \tag{4.1}$$

(5) 合成によって作成された画像に対して，物体の領域（基準座標 (x, y) と横・縦の大きさ (w, h)）・カテゴリを示したアノテーションファイルを作成する．この時，物体を合成した領域とそのカテゴリはシステムにとって既知であるため，自動的に作成できる．

上記，（4）～（5）をシステムが数万回繰り返すことでビックデータを生成する．

4.3 実験・考察

4.3.1 物体検出・認識の性能評価

提案手法と人手によりデータセットを作成し，それを DNN で訓練し，それぞれの物体検出・認識の性能を評価・比較した．まず，3 章で述べた手法に則り，提案手法を用いてデータセット（合成データ）を以下の手順で作成した．なお，提案手法に関わるパラメタは Table 4.1 に示す通りとした．

(1) 床・家具（机・棚・椅子など）を，RGB-D カメラ（ASUS Xtion Pro Live）で

撮影し 306 枚の背景画像を得た．また，それぞれの背景画像に対して物体が置かれるであろう点 p を GUI で指定した．

(2) Fig. 2.8 に示す WRC で使用された 15 個のおもちゃを，Fig. 4.2（2）に示す装置で撮影し，物体画像を得た．

(3) 背景画像と物体画像を合成し，同時にアノテーションファイルを得た．

これにより全 15600 枚の合成データを得て，その 15600 枚全てを合成訓練データとし，合成テストデータはなしとした．Fig. 6.2 に作成した合成データの一例を示す．なお，15600 枚の合成訓練データを得るのに要した時間は 2.5h であった．

Fig. 4.3　合成データの例

次に，提案手法と比較するため，2 章で述べた手法に則り，人手によりデータセット（実データ）を以下の手順で作成した．

(1) 床・家具（机・棚・椅子など）に，Fig. 2.8 に示す WRC で使用された 15 個のおもちゃをランダムな数選び，ランダムな位置・方向に置き，撮影した．

(2) 撮影した画像に対し，アノテーションファイルを記述した．

これにより，全 108 データ枚の実データを得て，その 54 枚を実訓練データとし，残りの 54 枚を実テストデータとした．なお，54 枚の実訓練データを得るのに要した時間は 2.5h で，合成訓練データを得る時間と揃えた．

Table 4.1 Parameters for experiments

Items	Values
d_{min}	1.0 m
d_{max}	2.5 m
h_{min}	1.0 m
h_{max}	1.3 m
θ_{min}	0.0 deg
θ_{max}	45.0 deg
m	2 cameras
d”	1.0 m
l	10 images
θ'_{min}	-10.0 deg
θ'_{max}	10.0 deg

また，これらの合成・実訓練データそれぞれを用いて，画像から End-to-end で物体検出・認識を同時に行う DNN である YOLO v2[12] を訓練した．YOLO v2 は，ImageNet [14]，COCO データセット [16] で学習したモデル・パラメタを 23 層目まで用い，それ以降を転移学習した．また，PC（Intel Core i7-8700K，DDR4 32GB，nVIDIA GTX 1080）で Darknet を用い，1 万エポック訓練した．それぞれの物体検出・認識の性能の評価は，実テストデータを真値として行った．

Table 4.2 データセット作成性能の比較

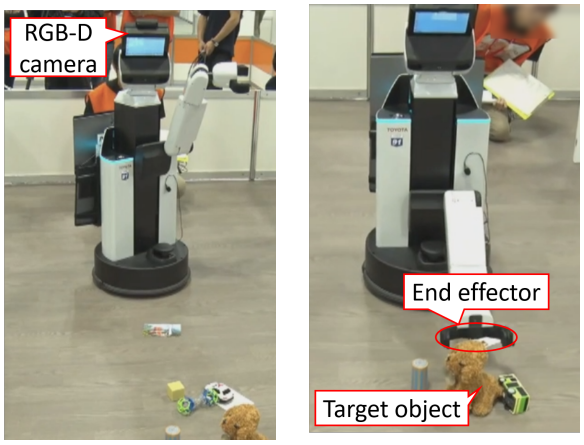
Evaluation index		Proposed method	Manpower
Number of data		15600	54
Work time	per 1 object [min/object]	4 to 12	N/A
	per 1 data [s/data]	0.58	166.67
	per 1 train dataset [h/dataset]	2.5	2.5
mAP		64.77	66.69

Fig. 4.4　物体検出・認識の結果

実験結果として，合成データを用いて，物体検出・認識した結果を，Fig. 4.4 に示す．また，合成データと実データを比較した結果を Table 4.2 に示す．この結果より，合成データは実データに比べ，物体検出・認識の性能を示す mAP が約 2 ポイント低下していることが確認できる．また，評価に用いた条件は異なるが，Georgakis らの手法と比べ合成データと実データの mAP の差が少なくなった．次に，合成データは実データに比べ，1 つのデータ作成に要する作業時間を，167s から 0.58s へ約 287 分の 1 に短縮した．ホームサービスロボットに応用することを考慮すると，データセット作成時の人間の作業負担が少ない提案手法が有効である．

4.3.2　ホームサービスロボットへの実装および評価

4.1 節で述べた合成訓練データで，DNN を訓練し，それを Toyota HSR に実装し，2.1.3 節で述べた WRC Tidy Up Stage1 を行い，以下に述べる手順で評価した．まず，ロボットは Fig. 4.5(a) に示すように，RGB-D カメラを 45deg チルトし床面の物体を検出・認識する．次に，ロボットは操作（把持）を試みる対象の物体名を宣言（発話）し，Fig. 4.5(b) に示すように対象物体までエンドエフェクタを制御する．この時，Fig. 4.5(b) に示すエンドエフェクタに，対象物体がわずかにでも触れたかどうかについて，ロボットの動作を撮影した映像から判定した．次に，ロボットは対象



(a)　(b)

Fig. 4.5　物体把持時のロボットの動作　(a) カメラを 45deg チルトし物体を探す　(b) 物体を把持する

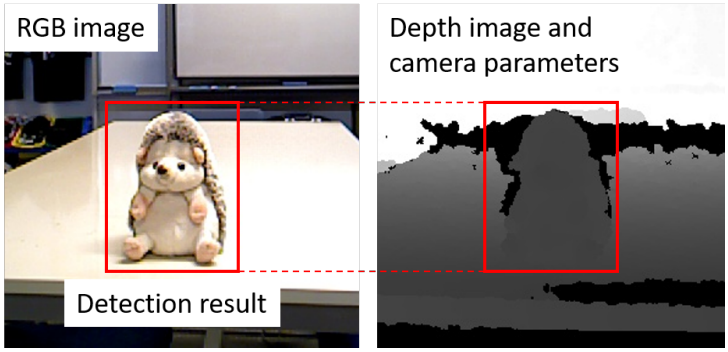


Fig. 4.6　物体検出領域，深度画像，カメラパラメタを用いた物体の 3 次元座標の推定

物体を持ち上げるためエンドエフェクタを制御する．この時，対象物体全てが床面から離れたかどうかについて，ロボットの動作を撮影した映像から判定した．なお，ロボットが把持を試みる目標座標は，物体検出・認識の後，Fig. 4.6 に示すように，物体検出領域（RGB 画像）と対応する Depth 画像の値およびカメラパラメタを用いて，物体検出領域のそれぞれのピクセルの 3 次元座標を求め，その平均とした．エンドエフェクタが対象物体に触れたか判定した結果を Fig. 4.7 に示す．Target object はロボットが宣言した対象物体で，Touched object はロボットが触れた物体を示す．全 10 回のうち，Fig. 4.7（1）～（8）の 8 回対象物体に触れることに成功した．Fig.

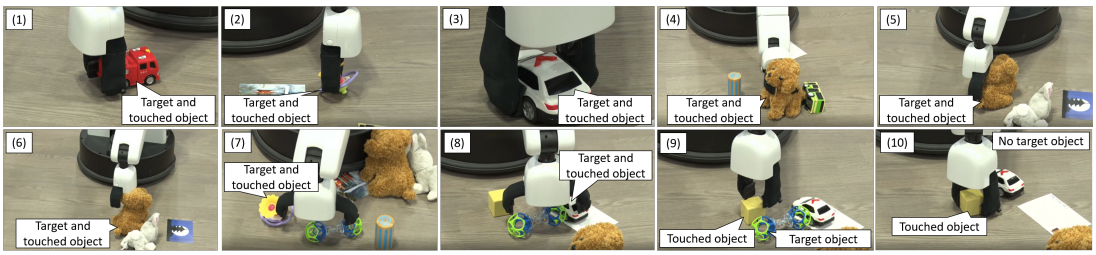


Fig. 4.7 物体に触れたかの判定に使用した画像

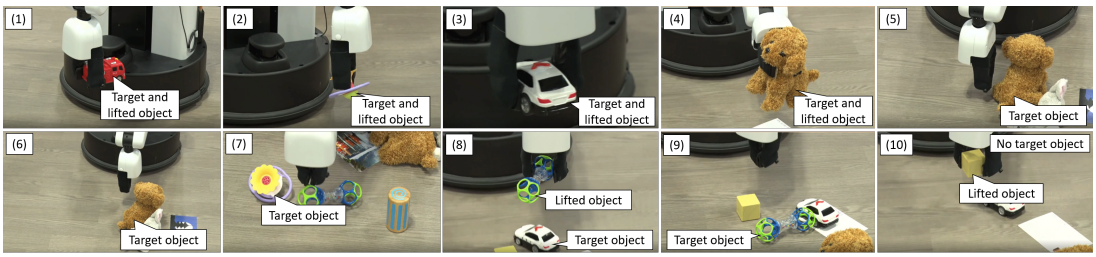


Fig. 4.8 物体を持ち上げたかの判定に使用した画像

4.7（9）は対象物体に隣接する他の物体に触れ失敗しており，Fig. 4.7（10）は対象物体が存在しない領域で他の物体に触れ失敗した．次に，対象物体が持ち上げられたか判定した結果を Fig. 4.8 に示す．Target object は前述の通りで，Lifted object はロボットが持ち上げ床面から離れた物体を示す．前述と同じ全 10 回のうち，Fig. 4.8（1）～（4）の 4 回対象物体を持ち上げることに成功した．Fig. 4.8（5）～（8）は対象物体に触れることができたが，持ち上げることに失敗した．Fig. 4.8（9）～（10）は対象物体に触れることが出来ず，持ち上げることにも失敗した．以上をまとめると，ロボットは全 10 回のうち，物体に触れることに 8 回成功し，物体の把持に 4 回成功した．その結果，WRC では 15 点を獲得し，全参加チーム内で最大の得点を得た．

今後の課題として，ホームサービスロボットへの提案手法の実装時において，物体操作の成功回数を向上する必要がある．そのため，物体にわずかに触れている場合（Fig. 4.7（7）（8）），および物体に触れていない場合（Fig. 4.7（9）（10））について，物体領域が正しく推論されているか検証する必要がある．また，物体に触れたが持ち上げられない場合（Fig. 4.8（5）（6））について，物体領域だけでなく，ロボットが把持すべき点も教師したデータセットを生成することが考えられる．

第 5 章

高位合成と ROS の統合

本章では，ロボットミドルウェアとアクセラレータを統合することを目標とする．前述した通り，ロボットミドルウェアとアクセラレータを結ぶ技術は未だ確立されていない．また，ロボットとアクセラレータ両分野にまたがる研究者は世界的に見ても希少で，これらを統合する学際領域研究は数が少ない．そこで，ロボットミドルウェアとアクセラレータを自動的に統合できるインタフェースおよびシステムを構築する．

5.1 関連研究

ロボットミドルウェアと FPGA を統合した研究例として Lange らの提案 [54] が挙げられる．この提案では，同研究グループの先行研究である Unity Link フレームワーク [55] を使用して，ROS と FPGA を繋ぐインタフェースを自動生成してる．Lange らは，このシステムの応用例として倒立振子による自律ナビゲーションを上げており，高レイテンシが要求されるアクチュエータ制御を FPGA に実装している．

この研究をホームサービスロボットや自動運転車の知的処理の効率化に生かすには，高レイテンシだけではなく高スループットなシステムに改良する必要がある．また，ROS と FPGA ではなく，ROS と hw/sw 複合体を繋ぐインタフェースの自動生成を行うことで更なる高効率を達成できる．これは，hw/sw 複合体にすることで，知的処理のみならず，センサのドライブといった負荷までオフロードできるためである．

ロボットミドルウェアと hw/sw 複合体を統合した研究例として，大川らの提案

[56, 57] が挙げられる．この研究は，FPGA を含めた ROS の Package を構築することを目指している．この研究では hw/sw 複合体として SoC を用いている．これは，CPU と FPGA を 1 チップに集積した計算機である．大川らは，SoC の CPU に ROS をインストールし，FPGA に知的処理を実装した．そして，ROS と FPGA のインタフェースを自動生成することにより，これら 2 つが協調動作が行えるシステムを提案した．

しかしながら，ホームサービスロボットや自動運転車において，計算機を SoC のみにすることは，計算負荷を考えると不可能である．このことから，大川らの提案を生かしたまま，SoC を PC にインストールされた ROS と統合するシステム提案しなければならない．

5.2　提案手法

筆者はホームサービスロボットのために“COMTA”（Connective Object for Middleware To Accelerator）というシステムを構築した．これは，ホームサービスロボットや自動運転車のようなロボット内の組み込み環境で，知的処理を高効率（高速かつ低消費電力）で行うアクセラレータである．

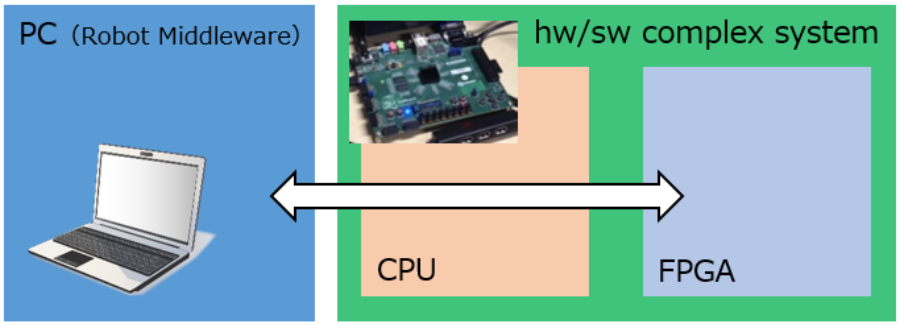


Fig. 5.1　COMTA の概要

Fig. 5.1 に COMTA の概要を示す．このシステムは，ロボットミドルウェアがインストールされた PC に，アクセラレータである hw/sw 複合体を加えた構成である．hw/sw 複合体は組み込み CPU と FPGA によって構成される．知的処理の効率化に hw/sw 複合体を用いるメリットは 2 点ある．1 点目は，FPGA により知的処理を高速，低消費電力で計算できる点である．これは関連研究で述べた通りで，FPGA は CPU，GPU に比べ計算速度，単位電力あたりの計算性能で勝っているためである．また，FPGA は内部回路を再構成可能なため，知的処理のアルゴリズムを変更することも容易である．2 点目は，組み込み CPU で既存のデバイスドライバを用いることが可能な点である．PC にセンサやアクチュエータを繋ぎ，データの交換を行うと，意外な程に計算資源を消費している．COMTA では，PC に代わり組み込み CPU がデータの交換を行うことで，システム全体で効率化を図る．

5.2.1　システム図を用いた COMTA の解説

次に，COMTA を用いた知的処理の効率化手法を解説する．COMTA を用いたシステムを構築する場合，前半を FPGA 技術者が，後半をロボット技術者が分業開発する．また，COMTA はそれぞれの技術者に Object と呼ぶ機能群を提供し，開発の効率化，システムの自動生成に寄与する．

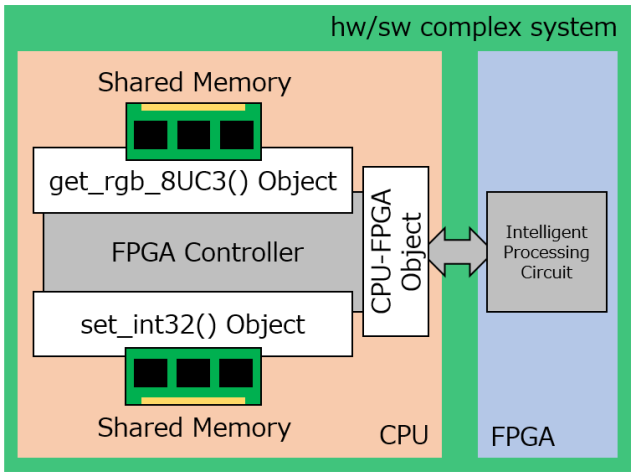


Fig. 5.2　FPGA 技術者が行う開発

Fig. 5.2 を用いて前半に FPGA 技術者が行う開発について解説する．まず, FPGA 技術者は図の灰色部分にあたる，FPGA への知的処理実装と，組み込み CPU への

FPGA コントローラ実装を行う．FPGA への知的処理実装は，最終的なシステムの効率を大きく左右するため，設計を熟知した FPGA 技術者が最適に実装しなければならない．FPGA コントローラも同じく，設計を熟知したものが実装しなければ FPGA の性能を殺すことになる．

前半において COMTA は，組み込み CPU と FPGA を繋ぐ Object を FPGA 技術者に提供する．これにより FPGA 技術者は組み込み CPU と FPGA を繋ぐインタフェースの開発・デバッグに掛かる工数を削減できる．また，COMTA は FPGA コントローラに対して，他とデータの交換を可能にする Object を提供する．この Object には，共有メモリを介したデータの交換口が実装されている．データ交換の手法を共有メモリベースにしている理由は，他の手法に比べ計算性能が低い組み込み CPU でも高速にデータの交換が行えるためである．

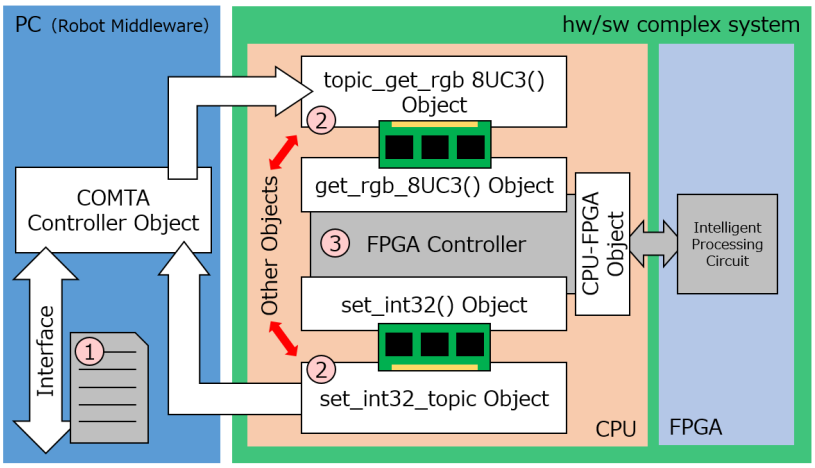


Fig. 5.3　ロボット技術者が行う開発

次に，Fig. 5.3 を用いて後半にロボット技術者が行う開発について解説する．最初に，ロボット技術者はロボットミドルウェアのインタフェースを介して hw/sw 複合体の振る舞いを COMTA コントローラに設定する（Fig. 5.3①）．この設定には，hw/sw 複合体でアクセラレートする知的処理，それに対する入出力が定義されている．次に，COMTA が設定に合わせ，データの入出力を行う Object を起動する（Fig. 5.3②）．最後に，FPGA とそのコントローラを起動する（Fig. 5.3③）．

これにより，ロボット技術者はロボットミドルウェアのインタフェースを操作するだけで，hw/sw 複合体を操作できる．そして，入出力データも全てロボットミドル

ウェアのインタフェースにより参照可能である．また，設定により hw/sw 複合体の振る舞いを変更することで，知的処理の種類や入出力データを変更でき，自由自在にシステム構成を変更可能である．

これらの前半，後半の開発は互いに完全に依存関係がない分業開発となっている．そして，前半，後半の開発は，COMTA が共有メモリベースでシステム実行時に自動的に繋ぎ合わせている．これにより，FPGA 技術者，ロボット技術者が互いの分野について無知な状態でも COMTA を使用しシステムを構築できる．FPGA の実装は，一般的なロボット技術者には馴染みのない HDL（Hardware Description Language）を用い，難易度が高い．また，FPGA に適した専用アルゴリズムを実装しなければならず，導入障壁が大きい．COMTA を用いることで，これらの障壁は取り除かれ，組み込み環境向け知的処理アクセラレータとして大きな波及効果が期待される．

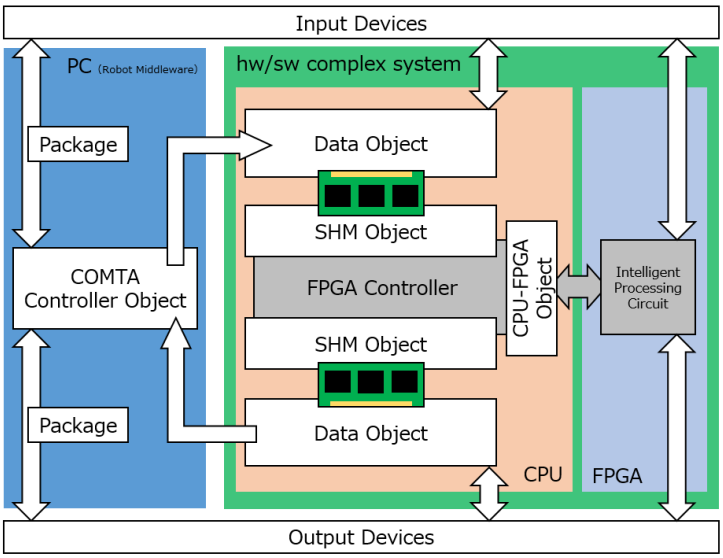


Fig. 5.4　COMTA システムブロック図

まとめると COMTA のブロック図は Fig. 5.4 の通りである．COMTA は以下の Object を技術者に提供する．

- 組み込み CPU と FPGA を繋ぐインタフェースを提供する CPU-FPGA Object
- FPGA コントローラに共有メモリベースのデータ交換機能を付ける SHM（SHared Memory） Object

- 入出力データを扱う Data Object
- ロボットミドルウェアから hw/sw 複合体を操作するための COMTA Controller Object

センサやアクチュエータなどは PC，組み込み CPU，FPGA 全てに接続可能な設計となっている．これは，知的処理に入出力するデータを柔軟に選択できるということである．（ただし，出来る限り組み込み CPU，FPGA で入出力を行うことが効率的といえよう．）

5.2.2　レイヤを用いた COMTA の解説

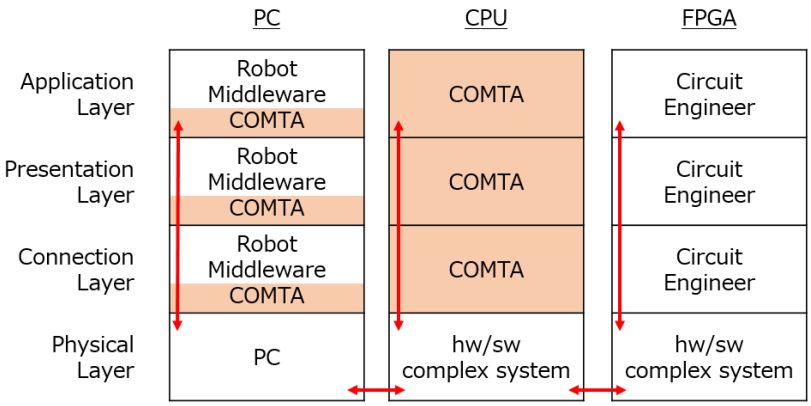


Fig. 5.5　COMTA が担う役割

Fig. 5.5 を用いて COMTA が担う役割を解説する．これは，レイヤ（機能に応じた段階構造）を定義した図である．それぞれのレイヤの意味は以下の通りである．

- アプリケーション層: アプリケーションごとの固有の規定
- プレゼンテーション層: データ表現形式の規定
- 通信層: データ交換の規定
- 物理層: 物理的な接続の規定

PC においては，ロボットミドルウェアがアプリケーション層から通信層まで管理を行う．FPGA においては，FPGA 技術者の実装によりアーキテクチャが決定され，アプリケーション層から通信層までカバーする．そして COMTA は，図の橙色部分

をカバーする．この部分は，従来であれば FPGA とロボット両方を熟知した技術者がシステムに合わせて 1 から実装していた．COMTA では，これを自動化し，また FPGA とロボット両方を熟知した技術者を不要とするため，ロボット内でのアクセラレータ使用に大きく貢献する．

5.3 実験・考察

5.3.1 提案システム（COMTA）の構築

提案システム（COMTA）を構築した．今回は，hw/sw 複合体として，Xilinx[58] の ZedBoard[59] を採用した．これは，Fig. 5.6 に示す，組み込み CPU と FPGA を 1 チップに集積した Zynq[60] の評価ボードである．ZedBoard では，組み込み CPU を PS（Processing System），FPGA を PL（Programmable Logic）と呼ぶ．

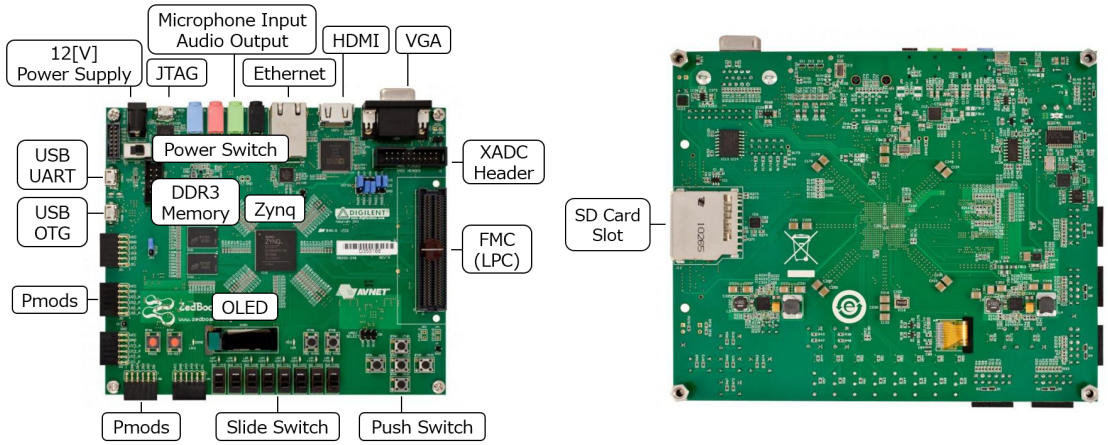


Fig. 5.6 ZedBoard

Table 5.1　ZedBoard の諸元

Chip	Zynq-7000 SOC XC7Z020-CLG484-1
CPU (PS)	Dual ARM Cortex-A9 MP Core (667MHz)
FPGA (PL)	55k Logic Cell 2.5Mb Block RAM 120 DSP
Memory	512MB DDR3 256Mb Quad-SPI Flash Full size SD/MMC card cage
Connectivity	10/100/1000Mb Etherenet USB OTG (Device/Host/OTG) USB UART
Power	DC 12V 3.0A (Max)
Size	About 160(H) × 135(W) mm

ZedBoard の諸元を Table 5.1 に示す．ZedBoard に搭載されている組み込み CPU はデュアルコアの ARM である．また搭載される FPGA のスペックは中間クラスである．ギガビット Ethernet が搭載されており，これを介して PC と通信する．最大消費電力は 36W と一般的性能のラップトップ PC のアイドリング時と大きく変わらない．サイズはホームサービスロボットや自動運転車への実装に適した小型である．

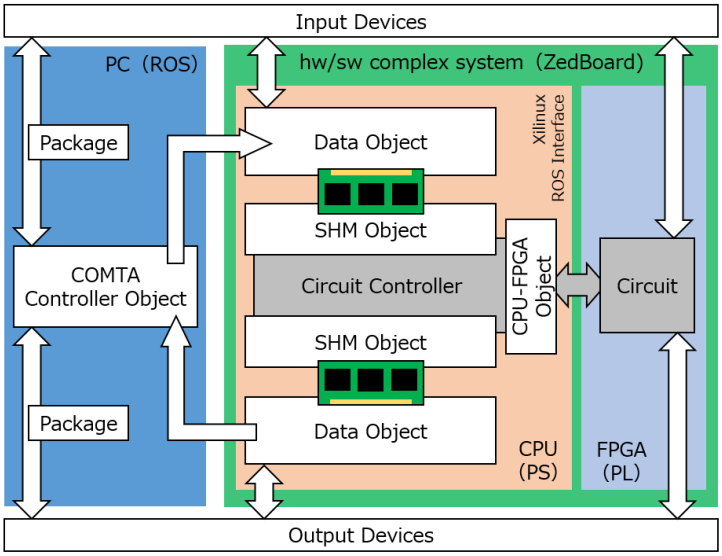


Fig. 5.7 構築したシステム（COMTA）

Fig. 5.7 に構築したシステム（COMTA）を示す．今回，PC で使用するロボットミドルウェアは ROS とした．ZedBoard の組み込み CPU には Xilinux[61] というOS をインストールし，ROS のインタフェースを実装した．

Table. 5.2 に構築した Object を示す．Data Object としてデバイスドライバにあたる 2 つの Object と，PC の ROS と画像や数値データの交換が可能な 5 つのObject を構築した．

Table 5.2 COMTA の Object

CPU-FPGA Object	
SHM Object	
Data Object	OpenNI Object
	V4L2 Object
	Img Object
	Int32 Object
	Int32MultiArray Object
	Float32 Object
	Float32MultiArray Object
COMTA Controller Object	

5.3.2　COMTA を用いた基礎実験

COMTA の基礎性能を評価する実験を行った．実験にあたり，評価の指標となるスループットとレイテンシについて解説する．

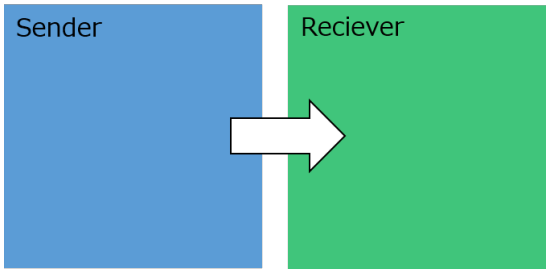


Fig. 5.8　スループット

Fig. 5.8 を用いてスループットについて解説する．スループットとは単位時間あたりに扱えるデータ量を指す．すなわち，通信の評価においては，送信者が受信者対してに単位時間あたりに送信可能なデータ量を指す．図中の，矢印の太さをスループットと考えると理解し易い．

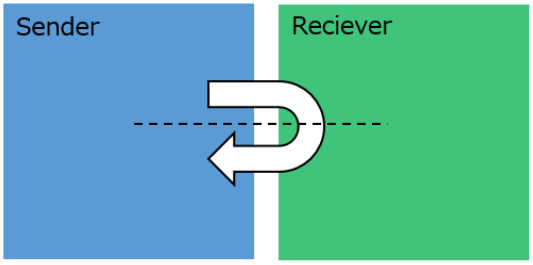


Fig. 5.9　レイテンシ

次に，Fig. 5.9 を用いてレイテンシについて解説する．レイテンシとは，データを入力してから出力するまでに要する時間を指す．すなわち，通信の評価においては，送信者が送ったしたデータが，何秒後に受信者によって受け取られるかを指す．しかしながら，送信者と受信者の間で時間を計測することは技術的に難しいため，送信者が送ったデータが，受信者を通り，また送信者へと戻ってくるまでの時間を計測し，その半分をレイテンシとすることが多い．

PC-組み込み CPU 通信の評価

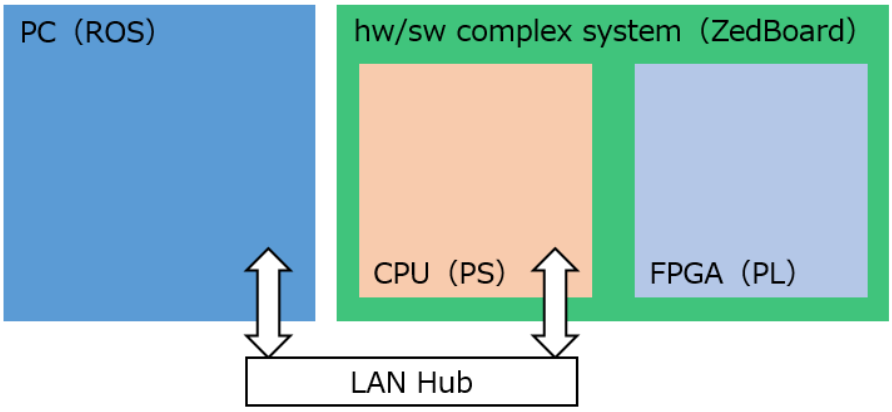


Fig. 5.10 PC と組み込み CPU 通信の評価環境

PC（ROS）-組み込み CPU（PS）通信を評価した．実験環境を Fig. 5.10 に示す．PC と組み込み CPU は LAN ハブ（BUFFALO LSW5-GT-8NS/BK）を介してギガビット Ethernet で接続した．この通信においては，スループット，レイテンシどちらの指標も重要となる．この通信では，データ量の大きい画像などの送受信が生じる．この時，データを滞りなく扱うため，高スループットな通信でなければならない．また，この通信では，ロボットを制御に直結するデータの送受信が生じる．この時，時間遅れの無いリアルタイムデータでロボットを制御するには，高レイテンシな通信でなければならない．本実験では，PC をデータ送信側とし実験を行った．

結果は，スループットが約 67.06MB/s，レイテンシが約 2.15ms であった．

組み込み CPU-FPGA 通信の評価

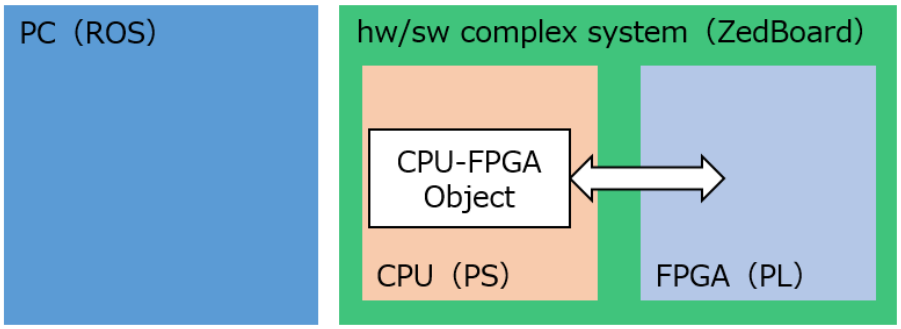


Fig. 5.11　組み込み CPU-FPGA 通信の評価環境

組み込み CPU（PS）-FPGA（PL）通信を評価した．実験環境を Fig. 5.11 に示す．この通信においては，スループット，レイテンシどちらの指標も重要となる．この通信では，データ量の大きい画像などの送受信が生じる．この時，データを滞りなく扱うため，高スループットな通信でなければならない．また，この通信では，ロボットを制御に直結するデータの送受信が生じる．この時，時間遅れの無いリアルタイムデータでロボットを制御するには，高レイテンシな通信でなければならない．本実験では，組み込み CPU と FPGA を繋ぐインタフェースを提供する CPU-FPGA Object をデータ送信側とし実験を行った．

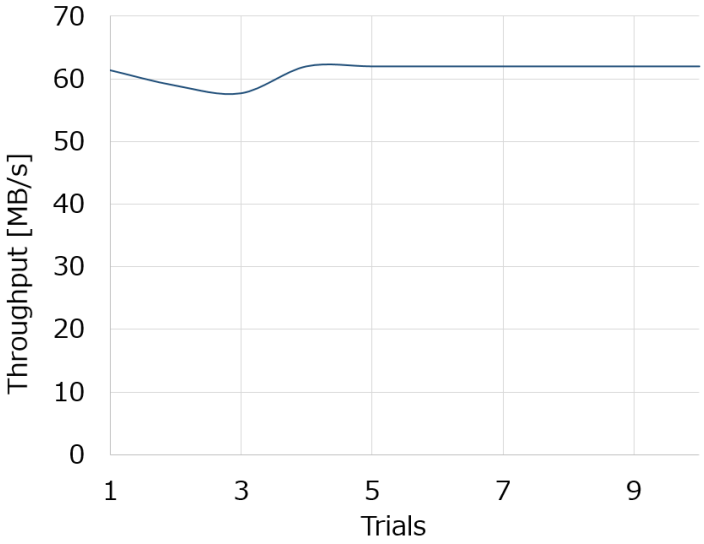


Fig. 5.12 組み込み CPU-FPGA スループット

Fig. 5.12 にスループットの結果を示す．グラフの縦軸はスループットを表し，横軸は試行回数で，10 回試行した結果を示す．スループットの平均は 61.26MB/s，分散は 2.19MB/s であった．

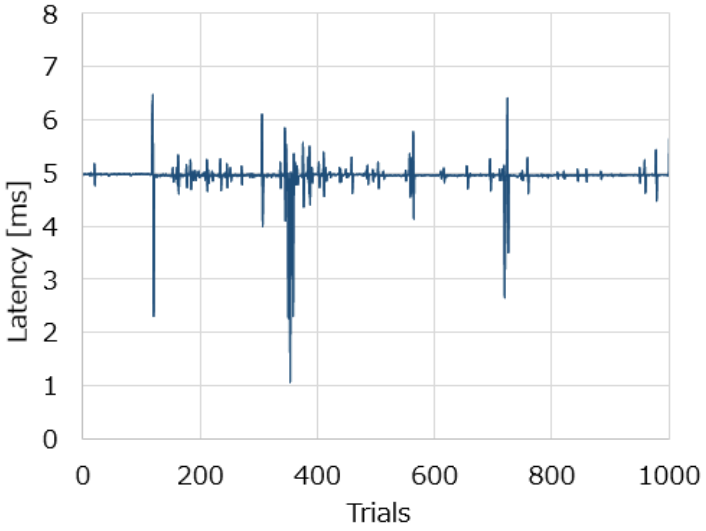


Fig. 5.13 組み込み CPU-FPGA レイテンシ

Fig. 5.13 にレイテンシの結果を示す．グラフの縦軸はレイテンシを表し，横軸は試行回数で，1000 回試行した結果を示す．レイテンシの平均は 4.94ms，分散は 0.06ms であった．

Message と共有メモリの比較

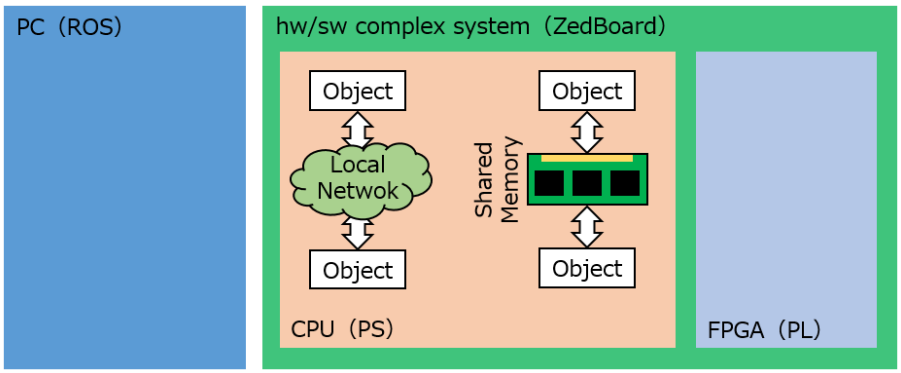


Fig. 5.14　組み込み CPU 内におけるデータ交換

組み込み CPU（PS）内のデータ交換において，ROS インタフェースである Message，提案手法である共有メモリどちらが有効か評価した．実験環境を Fig. 5.14 に示す．前者，Message を用いる手法は，組み込み CPU に ROS のインタフェースを実装しているため実現できる手法である．この手法では，組み込み CPU 内のローカルネットワークを経由してデータの交換が行われる．後者，共有メモリを用いる手法は，プロセス間データ通信によく用いられる手法である．本実験では，組み込み CPU 内に，2 つのデータ交換手法を実装した．また，それぞれに対し QVGA・VGA サイズの RGB 8 ビット 3 チャンネル画像を送受信し，フレームレートを計測した．

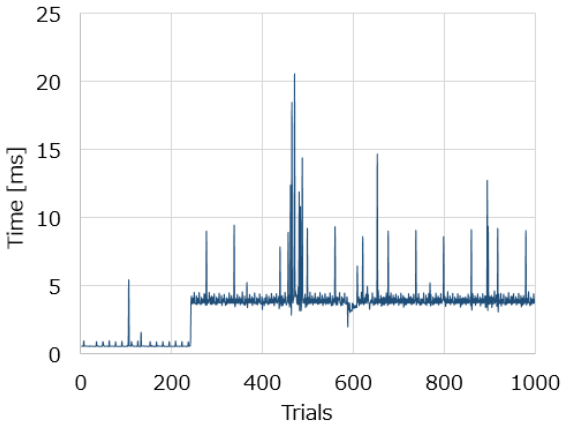


Fig. 5.15　Message 送信側 QVGA

Fig. 5.15 に QVGA サイズ，RGB 8 ビット 3 チャンネル画像を Message で送信（配信）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の送信に掛かった時間は平均 3.23ms，分散は 3.69ms であった．このことから，約 309.55fps のフレームレートで画像を送信できることが分かった．

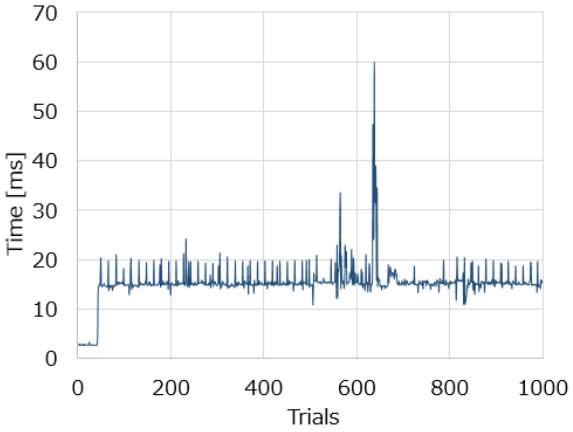


Fig. 5.16　Message 送信側 VGA

Fig. 5.16 に VGA サイズ，RGB 8 ビット 3 チャンネル画像を Message で送信（配信）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の送信に掛かった時間は平均 15.23ms，分散は 14.22ms であった．このことから，約 65.63fps のフレームレートで画像を送信できることが分かった．

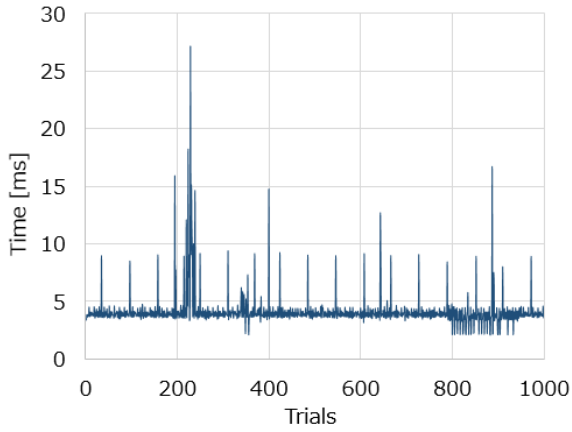


Fig. 5.17　Message 受信側 QVGA

Fig. 5.17 に QVGA サイズ，RGB 8 ビット 3 チャンネル画像を Message で受信（購読）した時の結果を示す．グラフの縦軸は画像 1 枚の受信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の受信に掛かった時間は平均 4.08ms，分散は 2.43ms であった．このことから，約 244.62fps のフレームレートで画像を受信できることが分かった．

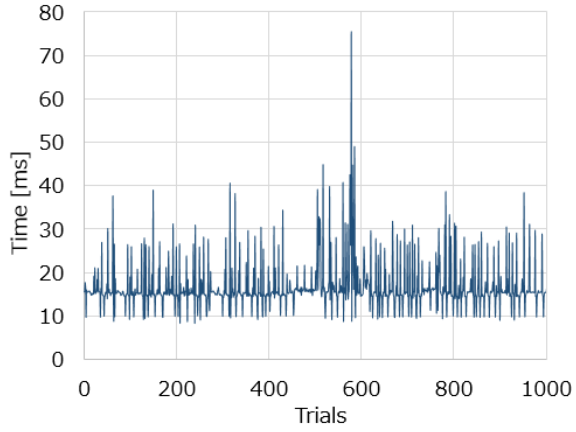


Fig. 5.18　Message 受信側 VGA

Fig. 5.16 に VGA サイズ，RGB 8 ビット 3 チャンネル画像を Message で受信（購読）した時の結果を示す．グラフの縦軸は画像 1 枚の受信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の受信に掛かった時間は平均 16.54ms，分散は 29.65ms であった．このことから，約 60.43fps のフレームレートで画像を受信できることが分かった．

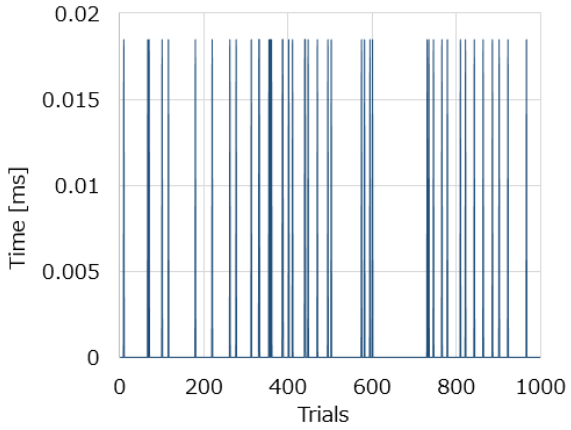


Fig. 5.19 共有メモリ 送信側 QVGA

Fig. 5.19 に QVGA サイズ，RGB 8 ビット 3 チャンネル画像を共有メモリに送信（書き込み）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の送信に掛かった時間は平均 0.79μs，分散は 13.98μs であった．このことから，約 1261703.88fps のフレームレートで画像を送信できることが分かった．

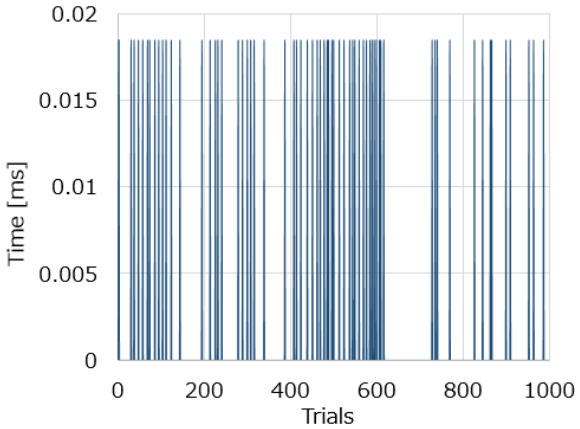


Fig. 5.20 共有メモリ 送信側 VGA

Fig. 5.20 に VGA サイズ，RGB 8 ビット 3 チャンネル画像を共有メモリに送信（書き込み）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の送信に掛かった時間は平均 1.22μs，分散は 20.94μs であった．このことから，約 822011.50fps のフレームレートで画像を送信できることが分かった．

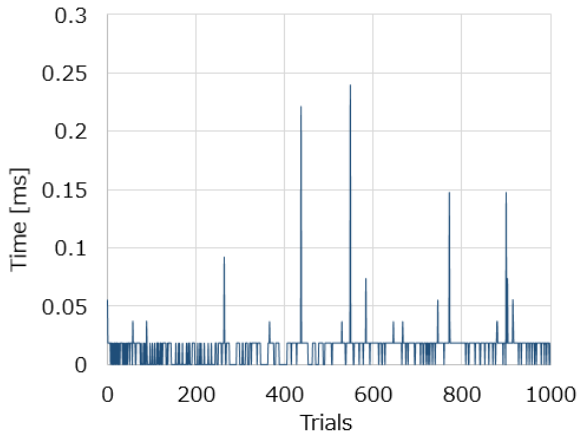


Fig. 5.21　共有メモリ 受信側 QVGA

Fig. 5.21 に QVGA サイズ，RGB 8 ビット 3 チャンネル画像を共有メモリから受信（読み込み）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の受信に掛かった時間は平均 0.01ms，分散は 0.22ms であった．このことから，約 70550.10fps のフレームレートで画像を受信できることが分かった．

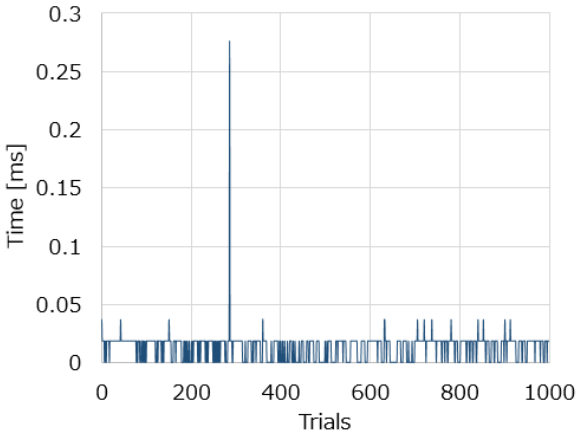


Fig. 5.22　共有メモリ 受信側 VGA

Fig. 5.22 に VGA サイズ，RGB 8 ビット 3 チャンネル画像を共有メモリから受信（読み込み）した時の結果を示す．グラフの縦軸は画像 1 枚の送信に掛かった時間，横軸は試行回数で，画像を 1000 枚送信した結果を示す．画像 1 枚の受信に掛かった時間は平均 0.01ms，分散は 0.15ms であった．このことから，約 77393.80fps のフレームレートで画像を取得できることが分かった．

Table 5.3 Message と共有メモリの比較結果まとめ

Method	Send/Recieve	Ave. Frame Rate [fps]	
		QVGA	VGA
Message	Send	309.55	65.63
	Recieve	244.62	60.43
Shared Memory	Send	1261703.88	822011.50
	Recieve	70550.10	77393.80

この実験の結果を Table 5.3 にまとめる．この表より，QVGA，VGA どちらのサイズにおいても，Message に比べ共有メモリの方が，送受信ともに高速であることが分かる．

システム再構成の確認

本実験では，ROS のインタフェースによりシステムの再構成が可能か確認した．Fig. 5.23 に示すように，COMTA Controller Object に対して ROS のインタフェースを介して設定を与えることによりシステムの再構成が可能か評価した．この再構成とは，Object の入れ替えにより，システムの振る舞いを変更することを指す．

本実験では，最初に Data Object の Float32 Object（get_float_32_topic），Int32 Object（get_int32_topic）を ROS のインタフェースにより起動した．次に，Float32 Object（get_float_32_topic）の終了，Float32 Object（get_float32_multi_array_topic）の起動が ROS インタフェースにより行えるか確認した．

結果を Fig. 5.24 に示す．手順通り，システムを再構成出来たことが分かる．

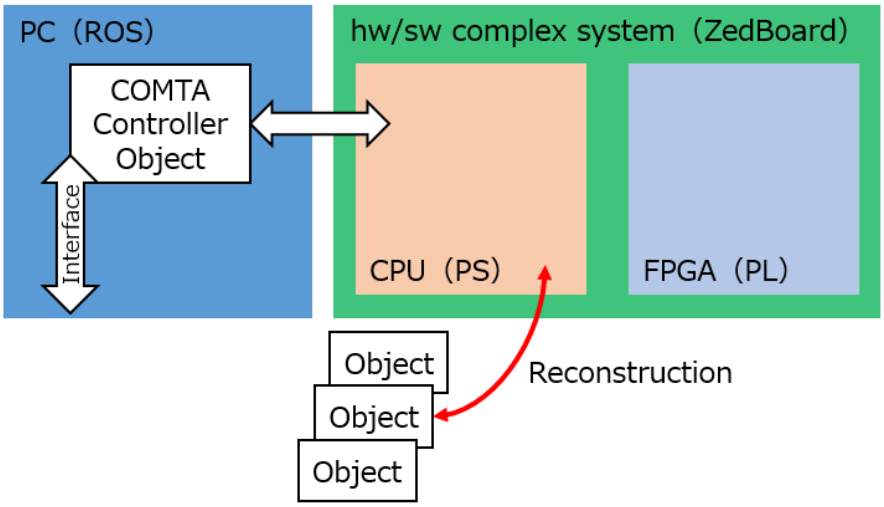


Fig. 5.23　システム再構成の評価

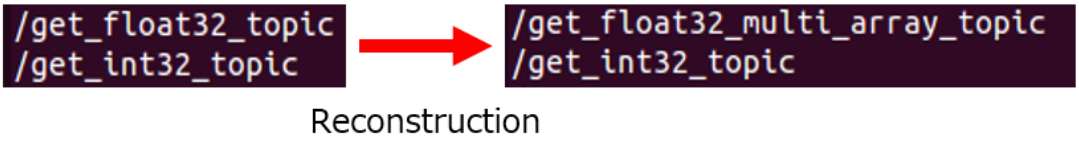


Fig. 5.24　システム再構成の結果

5.3.3 COMTA を用いた応用実験

人物検出のソフトウェア実装

ここまでの実験で，COMTA の基礎性能を評価した．ここからは，COMTA にアクセラレートする知的処理を実装していく．

本実験では，画像処理による人物検出を実装対象とした．人物検出は，ホームサービスロボットや自動運転車において，欠かせない知的処理の 1 つである．ホームサービスロットでは人間と自然なインタラクションをするために人物検出が必要である．また，自動運転車では歩行者検出を行い，安全運転を行うために必要である．さらに，画像処理は FPGA が得意としている信号処理の 1 つである．

今回は人物検出のアルゴリズムとして，Iwata らが提案している，MRCoHOG（Multi-Resolution Co-occurrence Histogram of Oriented Gradients）特徴量と RealAdaBoost 認識器を用いた人物検出 [64] を参考にした．

MRCoHOG 特徴量とは，HOG 特徴量 [5] の改良手法として提案されたものである．これらは，局所領域に着目することで，人間の姿勢など幾何学的変化に対応した特徴量である．また，局所領域の輝度勾配の頻度分布を作成することで，色や環境光の変化に頑健である．これらの特徴量は，物体の大まかな形を表すことが出来る．

Real Adaboost は，学習サンプルの中から，認識に有効な特徴量を逐次的に選択する統計学的手法である．これにより認識を行うときは，選択された少量の特徴量のみを用いればよい．これは，特徴量抽出段階，認識段階どちらにおいても計算量の削減に繋がり，高速化が図れる．

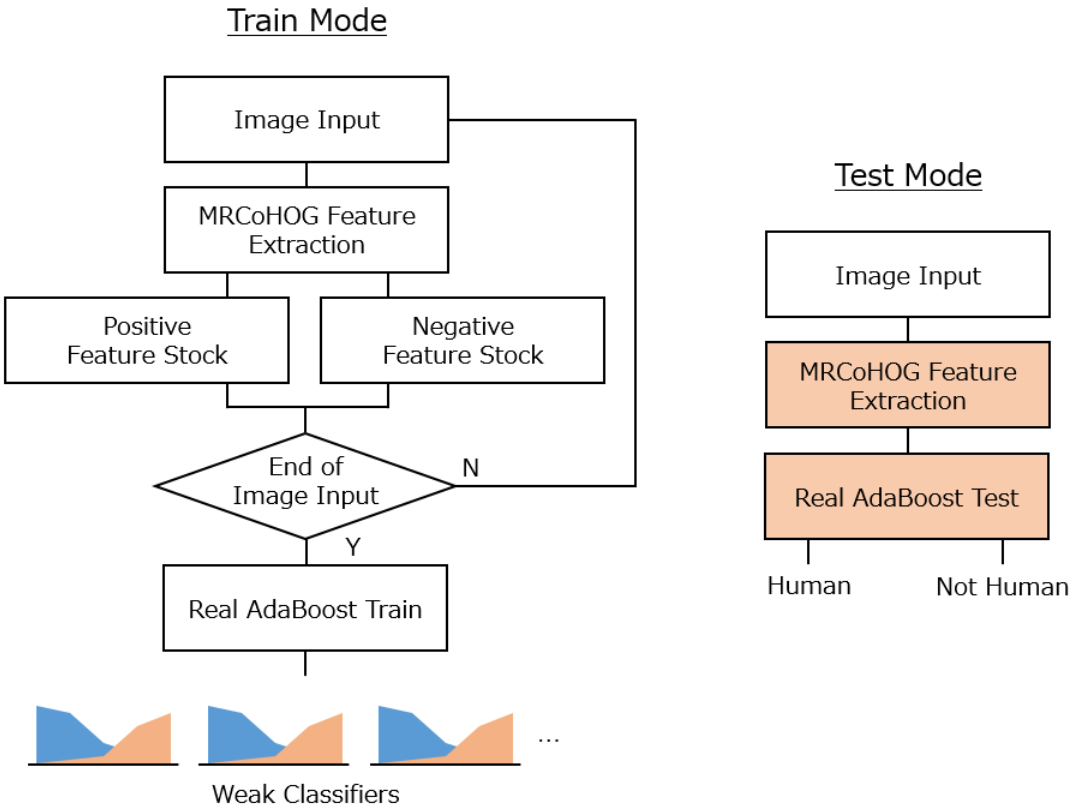


Fig. 5.25　人物検出アルゴリズム

まず，PC を用いて人物検出をソフトウェア実装し，アルゴリズムの検証を行った．（ノイマン型アーキテクチャの計算器に実装することをソフトウェア実装という）．Fig. 5.25 に人物検出アルゴリズムを示す．これは，図左に示す学習モードと，図右に示す検証モードに分けられる．

学習モードでは，グレースケール画像を入力とし，MRCoHOG 特徴量を取得する．この特徴量は，Positive（人物）と Negative（非人物）に分けてストックされる．全画像の入力が終了すると，ストックした特徴量を学習状態の RealAdaBoost に与える．学習結果は，学習回数に応じた数の弱識別器（ヒストグラム）として出力される．

検証モードでは，グレースケール画像を入力とし，MRCoHOG 特徴量を取得する．この特徴量は，実行状態の RealAdaBoost に与えられる．RealAdaBoost は学習状態で出力された弱識別器を用いて，認識結果を人物か非人物かの 2 値で出力する．

Fig. 5.26　人物検出データセット

今回は，RealAdaBoost に Fig. 5.26 に示すデータセットを学習させた．これは，INRIA データセット [66] から Positive（人物）と Negative（非人物）を抜き出したものである．本実験で設定したパラメータは Table. 5.4 の通りである．

Table 5.4　人物検出のパラメータ

MRCoHOG	Size of Original Image	64(H) × 32(W) pix
	Size of Resized Image 1	32(H) × 16(W) pix
	Size of Resized Image 2	16(H) × 8(W) pix
	Size of Block	8(H) × 8(W) pix
	Offset Distance	1
	Number of Gradient Direction	8
	Threshold of Gradient Magnitude	15.0
Real AdaBoost	Number of Histgram Bin	32
	Trials of Train	250
Dataset	Train Positive (Human)	2416 Images
	Train Negative (Not Human)	12288 Images
	Test Positive (Human)	1126 Images
	Test Negative (Not Human)	4840 Images

Table 5.5　ソフトウェア実装した人物検出の認識結果

		Input	
		Positive	Negative
Predict	Positive	1014	69
	Negative	112	4771

Table. 5.5 にソフトウェア実装による人物検出の認識結果をまとめる．これは，検証モードに Positive（人物）画像を 1126 枚，Negative（非人物）画像を 4840 枚入力し，認識結果をまとめたものである．Positive 画像を入力し，Positive 画像と認識されたのは，1014 枚であった．これを TP（True Positive）という．反対に，Positive 画像を入力し，Negative 画像と認識されたのは，112 枚であった．これを FN（False Negative）という．また，Negative 画像を入力し，Negative 画像と認識されたのは，4771 枚であった．これを TN（True Negative）という．反対に，Negative 画像を入力し，Positive 画像と認識されたのは，69 枚であった．これを FP（False Positive）という．これらの値から正解率を式 5.1 から求めると，96.97% となった．

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{5.1}$$

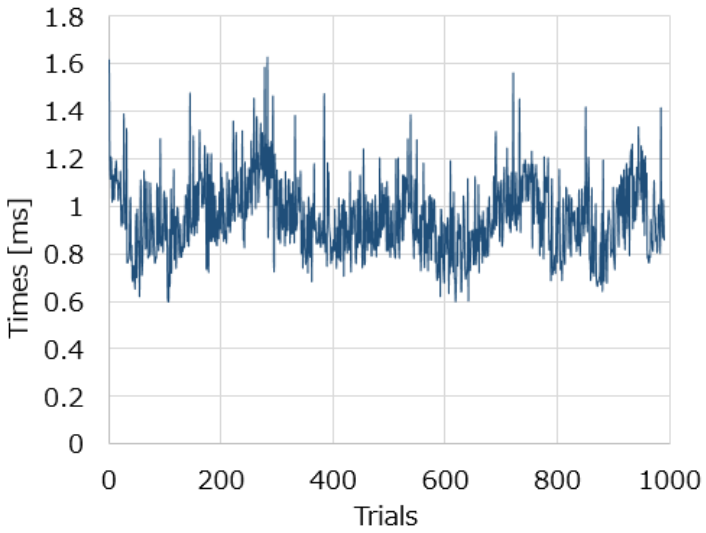


Fig. 5.27　PC による人物検出処理速度

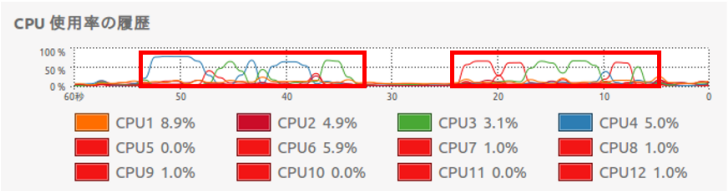


Fig. 5.28　PC に対する人物検出の負荷

Fig. 5.27 はソフトウェア実装した人物検出を PC で実行した時の処理速度である．これは，検証モードで画像入力から推測結果の出力までに要した時間（1 回の人物検出に要した時間）をまとめたものである．グラフの縦軸は，1 回の人物検出に要した時間，横軸は試行回数で，1000 回試行した結果を示す．今回用いた PC は，Intel Core-i7 6800K 3.4GHz 6 コア 12 スレッド，DDR4 32GB である．

1 回の人物検出に要した時間の平均は 0.95ms で，分散は 0.03ms であった．このことから，1 秒あたり約 1046.30 回の人物検出が行えることが分かった．また，PC の消費電力をワットメータ（REVEX ET30D）を用いて計測したところ，約 73W であった．これらから PC では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 14.33 回人物検出が行えることが分かった．

Fig. 5.28 は PC で人物検出実行時の CPU の使用率である．赤で囲った部分が実行中で，1 スレッドで 60% 程度の使用率が確認された．

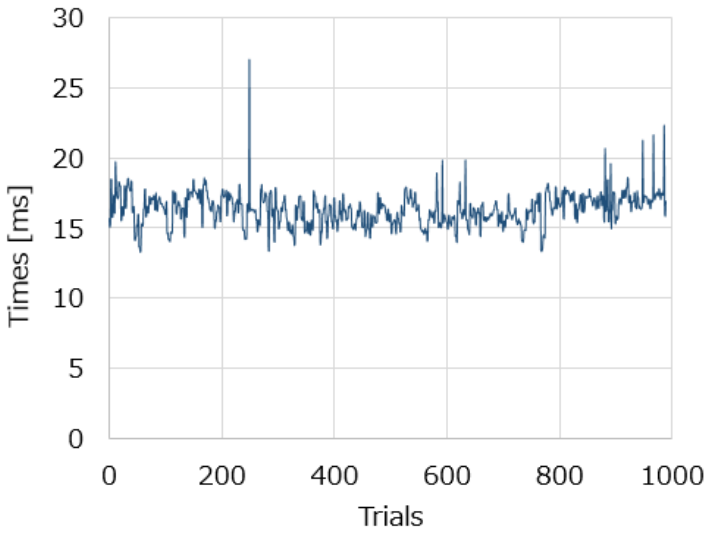


Fig. 5.29　組み込み CPU による人物検出速度

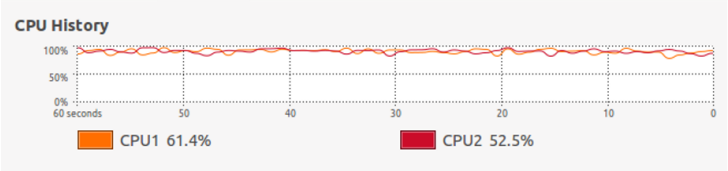


Fig. 5.30　組み込み CPU に対する人物検出の負荷

Fig. 5.29 はソフトウェア実装した人物検出を ZedBoard の組み込み CPU（PS）で実行した時の処理速度である．これは，検証モードで画像入力から推測結果の出力までに要した時間（1 回の人物検出に要した時間）をまとめたものである．グラフの縦軸は，1 回の人物検出に要した時間，横軸は試行回数で，1000 回試行した結果を示す．

1 回の人物検出に要する時間の平均は 16.53ms で，分散は 1.52ms であった．このことから，1 秒あたり約 60.50 回の人物検出が行えることが分かった．また，ZedBoard の消費電力を計測したところ，約 4.68W であった．これらから組み込み CPU では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 12.93 回人物検出が行えることが分かった．

Fig. 5.30 は組み込み CPU で人物検出実行時の CPU の使用率である．2 コア共に 80% 程度の使用率が確認された．

人物検出のソフトウェア・ハードウェア複合実装

ソフトウェア実装した人物検出を，ZedBoard の組み込み CPU（PS) と FPGA（PL）を用いてソフトウェア・ハードウェア複合実装した．（FPGA など非ノイマン型アーキテクチャの計算機に実装することをハードウェア実装という）．

本実験でソフトウェア・ハードウェア複合実装したのは，Fig. 5.25 の右に示す検証モードである．この中で，ソフトウェア実装したのは白色部分，ハードウェア実装したのは橙色部分である．

ハードウェア実装するにあたり，MRCoHOG 特徴量抽出はパイプライン化した専用アルゴリズムに，RealAdaBoost は LUT（Look Up Table）と加算器を用いた専用アルゴリズムに変更した．

ハードウェア実装には高位合成環境である Xilinx の Vivado HLS 2016.2[67] を用いた．これは，C 言語や C++ 言語で記述されたアルゴリズムを HDL に変換する環境である．また，HDL から bitstream ファイルの生成は，Xilinx の Vivado 2015.4[68] を用いた．Table 5.6 にハードウェア実装のレイテンシ，Table 5.7 にハードウェア実装の資源使用率を示す．

Table 5.6 ハードウェア実装のレイテンシ

Latency	Min	Max
[ns]	257390	595310

Table 5.7 ハードウェア実装の資源使用率

Resource	Available	Used	Untilization
BRAM_18K	280	0	0%
DSP_48E	220	0	0%
FF	106400	12822	12%
LUT	53200	20986	39%

Table 5.8　ソフトウェア・ハードウェア複合実装した人物検出の認識結果

		Input	
		Positive	Negative
Predict	Positive	1021	90
	Negative	105	4750

Table 5.8 にソフトウェア・ハードウェア複合実装した人物検出の認識結果をまとめる．これは，検証モードに Positive（人物）画像を 1126 枚，Negative（非人物）画像を 4840 枚入力し，認識結果をまとめたものである．Positive 画像を入力し，Poisitive 画像と認識されたのは，1021 枚であった（TP）．反対に，Positive 画像を入力し，Negative 画像と認識されたのは，105 枚であった（FN）．また，Negative 画像を入力し，Negative 画像と認識されたのは，4750 枚であった（TN）．反対に，Negative 画像を入力し，Positive 画像と認識されたのは，90 枚であった（FP）．これらの値から正解率を式 5.1 から求めると，96.73% となった．

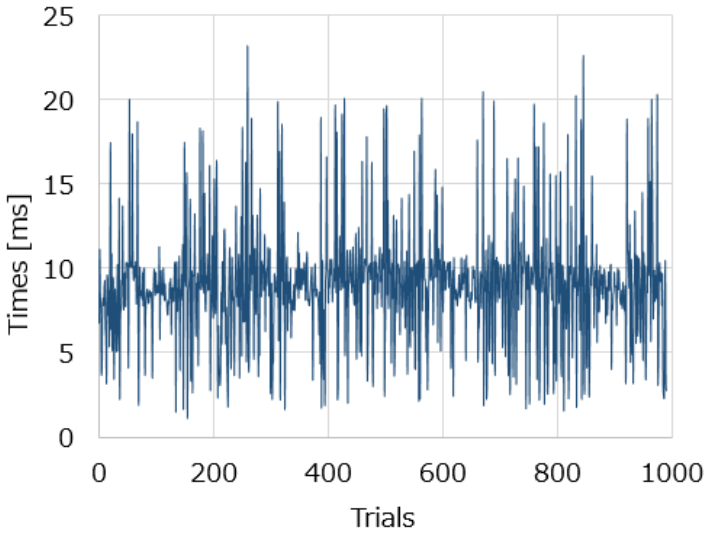


Fig. 5.31　組み込み CPU・FPGA による人物検出処理速度

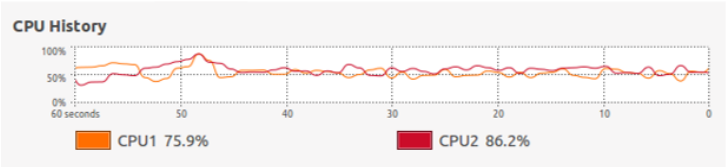


Fig. 5.32　組み込み CPU・FPGA に対する人物検出の負荷

Fig. 5.31 はソフトウェア・ハードウェア複合実装した人物検出を ZedBoard の組み込み CPU（PS）・FPGA（PL）で実行した時の処理速度である．これは，画像入力から推測の出力までに要した時間（1 回の人物検出に要した時間）をまとめたものである．グラフの縦軸は，1 回の人物検出に要した時間，横軸は試行回数で，1000 回試行した結果を示す．

1 回の人物検出に要した時間の平均は 8.68ms で，分散は 8.80ms であった．このことから，1 秒あたり約 115.24 回の人物検出が行えることが分かった．また，ZedBoard の消費電力を計測したところ，約 4.68W であった．これらから，組み込み CPU・FPGA では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 24.62 回人物検出が行えることが分かった．

Fig. 5.32 は組み込み CPU・FPGA で人物検出実行時の CPU の使用率である．2 コア共に 60% 程度の使用率が確認された．

Table 5.9　人物検出の結果まとめ

	DPS [dps]	DPS Per Watt [dpspw]
PC	1046.30	14.33
ZedBoard（CPU）	60.50	12.93
ZedBoard（CPU + FPGA）	115.24	24.62

この実験の結果を Table 5.9 にまとめる．ここで，DPS（The number of Detection Per Second）とは 1 秒あたりの人物検出回数を指す．この表より，単位電力あたりの計算性能を評価すると，ZedBoard の組み込み CPU（PS）・FPGA（PL）でソフトウェア・ハードウェア複合実装した結果が最も良いことが分かる．

人物追跡のソフトウェア実装

ここまでは，人物検出をソフトウェア・ハードウェア複合実装した．ここからは，人物検出を従来 Exi@に実装されていた人物追跡と統合し，実証実験を行う．

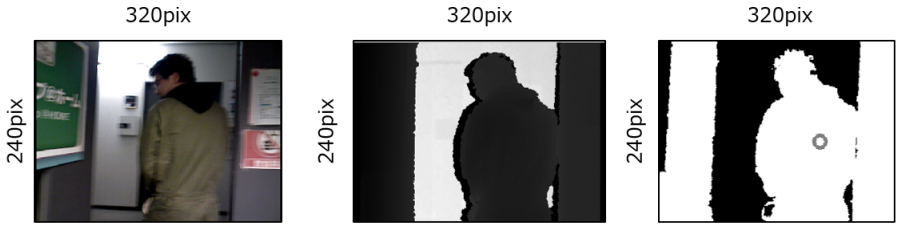


Fig. 5.33　従来の人物追跡アルゴリズム

最初に，Fig 5.33 を用いて従来の人物追跡アルゴリズムを解説する．図左は，RGB-D カメラから取得された QVGA サイズのカラー画像である．図中は，同じく QVGA サイズの深度画像である．図右は，追跡処理後の画像である．従来の人物追跡は，深度画像とパーティクルフィルタを用いた実装となっている．図右に示すよう，深度画像において一定距離を示すピクセルを白（255），それ以外を黒（0）として 2 値画像にする．これに対し，画像の白領域を確率的に探索するパーティクルフィルタを用いて，追跡を行う．従来の人物追跡は，深度情報のみを考慮していたため，図右のように人間以外の領域が白になることがあり，誤追跡をすることがある．この問題を解決するため，本実験では従来の人物追跡に人物検出を追加する．

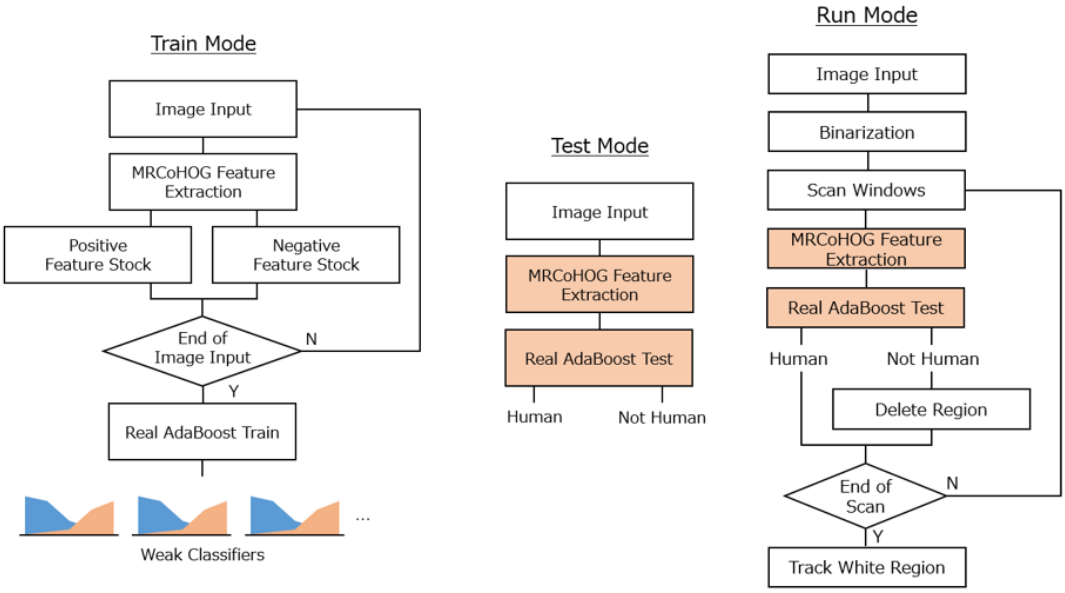


Fig. 5.34　改良型人物追跡アルゴリズム

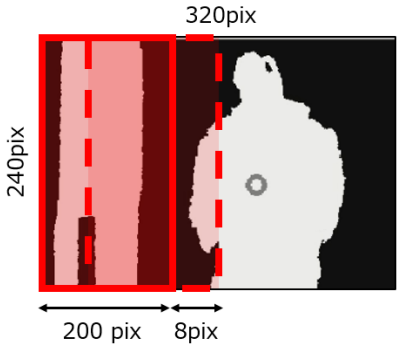


Fig. 5.35　2 値画像に対する検出ウィンドウ

Fig. 5.34 に改良型人物追跡アルゴリズムを示す．これは，人物検出と同様に学習モードと検証モードが存在し，さらに実行モードが加わっている．学習モード，検証モードのアルゴリズムに変更はない．この 2 モードを通して，Positive（人物），Negative（非人物）の認識が高精度に行える弱識別器を生成する．

実行モードは，まず従来通り 2 値画像を生成する．次に，Fig. 5.35 に示すように，2 値画像に対し検出ウィンドウを走査する．この時，各検出ウィンドウに対して MRCoHOG 特徴量抽出と RealAdaBoost による認識を行う．これにより，Negative（非人物）と判断された領域は白（255）から灰，黒（254〜0）へと値を減らしていき，人の領域のみが白（255）で残るようにする．最後に，従来通り，白（255）の領域をパーティクルフィルタにより追従することで人物追跡を行う．

Fig. 5.36 人物追跡データセット

今回は，RealAdaBoost に Fig. 5.36 に示すデータセットを学習させた．これは，Fig. 5.35 に示す検出ウィンドウから抜き出された画像を，Positive（人物）と Negative（非人物）に分けたものである．本実験で設定したパラメータは Table 5.4 の通りである．

Table 5.10 人物追跡のパラメータ

MRCoHOG	Size of Original Image	64(H) × 32(W) pix
	Size of Resized Image 1	32(H) × 16(W) pix
	Size of Resized Image 2	16(H) × 8(W) pix
	Size of Block	8(H) × 8(W) pix
	Offset Distance	1
	Number of Gradient Direction	8
	Threshold of Gradient Magnitude	15.0
Real AdaBoost	Number of Histgram Bin	32
	Trials of Train	100
Dataset	Train Positive (Human)	1465 Images
	Train Negative (Not Human)	10910 Images
	Test Positive（Human）	200 Images
	Test Negative（Not Human）	1250 Images

Table 5.11　ソフトウェア実装した人物追跡の認識結果

		Input	
		Positive	Negative
Predict	Positive	195	7
	Negative	5	1243

Table 5.11 にソフトウェア実装による人物追跡の認識結果をまとめる．これは，検証モードに Positive（人物）画像を 200 枚，Negative（非人物）画像を 1250 枚入力し，認識結果をまとめたものである．Positive 画像を入力し，Positive 画像と認識されたのは，195 枚であった（TP）．反対に，Positive 画像を入力し，Negative 画像と認識されたのは，5 枚であった（FN）．また，Negative 画像を入力し，Negative 画像と認識されたのは，1243 枚であった（TN）．反対に，Negative 画像を入力し，Positive 画像と認識されたのは，7 枚であった（FP）．これらの値から正解率を式 5.1 から求めると，99.17% となった．

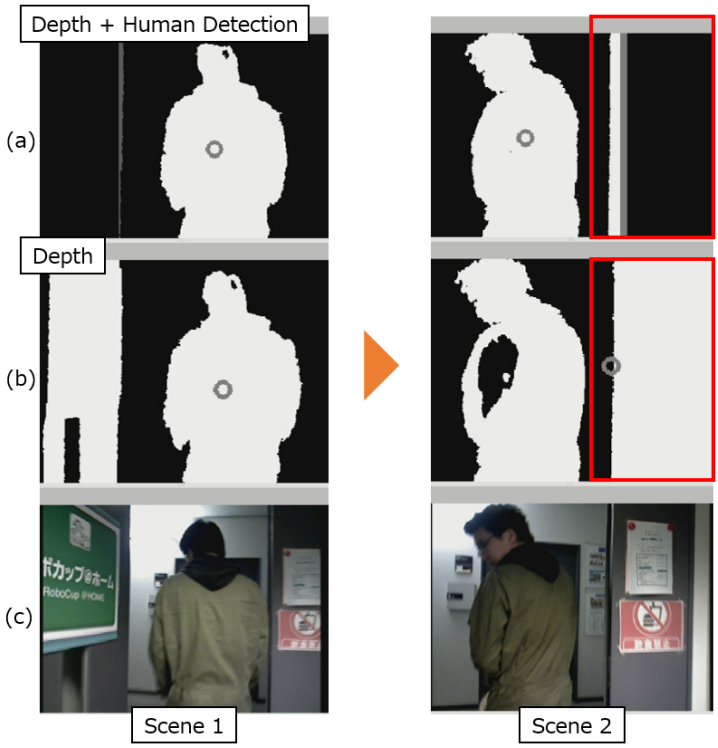


Fig. 5.37　従来手法との比較．（a）改良型（b）従来（c）カラー画像

Fig. 5.37 に従来の人物追跡（深度情報のみ）と，改良型人物追跡（深度情報と人物検出）の比較を示す．図左の段階では，従来の人物追跡では壁領域が白になっており，改良型人物追跡では黒になっている．このことから，改良型人物追跡では人間以外の領域を正しく削除出来ていることが分かる．また，この段階では図の灰色丸に示すように，どちらも問題なく追跡を続けている．図右の段階でも同様に，従来の人物追跡では壁領域が白になっており，改良型人物追跡では黒になっている．このため，従来の人物追跡では接近した壁領域に追跡対象が移動し，誤追跡をしている．これに対し，改良型人物追跡では，接近した壁領域に影響を受けることなく追跡を続けている．

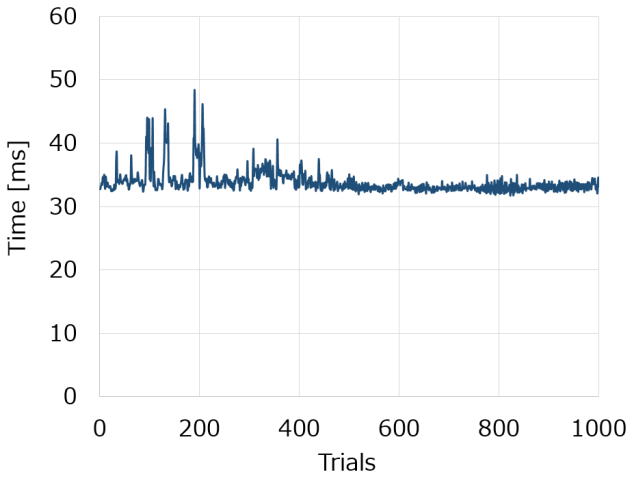


Fig. 5.38　PC による人物追跡処理速度（無負荷状態）

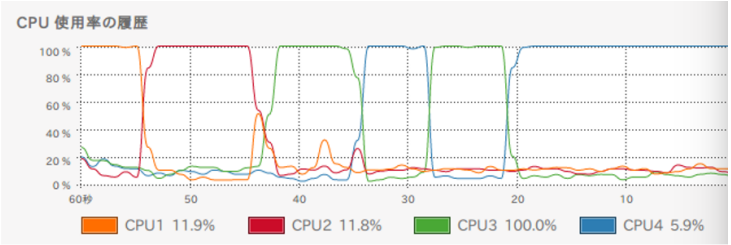


Fig. 5.39　PC に対する人物追跡の負荷（無負荷状態）

Fig. 5.38 はソフトウェア実装した改良型人物追跡を無負荷状態の PC で実行した時の処理速度である．これは，実行モードで画像入力から追跡座標の出力までに要した時間（1 回の追跡座標の更新に要した時間）をまとめたものである．グラフの縦軸は，1 回の追跡座標の更新に要した時間，横軸は試行回数で，1000 回試行した結果を示す．今回用いた PC は，Exi@に搭載される Intel Core-i5 5200U 2.2GHz 2 コア 4 スレッド，DDR3 12GB である．また，PC は RGB-D カメラドライバと人物追跡のみを実行し，他に計算負荷がない状態（無負荷状態）で計測した．

1 回の追跡座標の更新に要した時間の平均は 34.41ms で，分散は 5.10ms であった．このことから，1 秒あたり約 29.06 回の追跡座標の更新が行えることが分かった．また，PC の消費電力をワットメータ（REVEX ET30D）を用いて計測したところ，約 26W であった．これらから，PC では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 1.12 回の追跡座標の更新が行えることが分かった．

Fig. 5.39 は無負荷状態の PC で人物追跡実行時の CPU の使用率である．3 スレッドが 10% 程度，1 スレッドが 100% 程度の使用率が確認された．

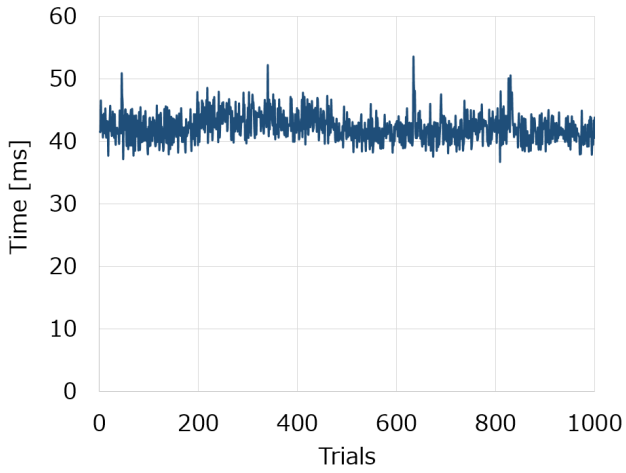


Fig. 5.40　PC による人物追跡処理速度（負荷状態）

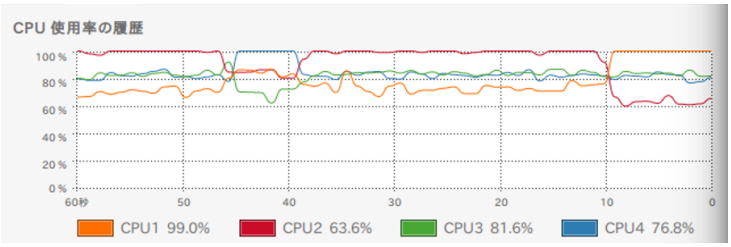


Fig. 5.41　PC に対する人物追跡の負荷（負荷状態）

Fig. 5.40 はソフトウェア実装した改良型人物追跡を負荷状態の PC で実行した時の処理速度である．これは，実行モードで画像入力から追跡座標の出力までに要した時間（1 回の追跡座標の更新に要した時間）をまとめたものである．グラフの縦軸は，1 回の追跡座標の更新に要した時間，横軸は試行回数で，1000 回試行した結果を示す．今回用いた PC は，Exi@に搭載される Intel Core-i5 5200U 2.2GHz 2 コア 4 スレッド，DDR3 12GB である．また，PC は RGB-D カメラドライバと人物追跡以外に，Exi@起動時に稼働するソフトウェアを実行した状態（負荷状態）で計測した．

1 回の追跡座標の更新に要した時間の平均は 42.08ms で，分散は 5.35ms であった．このことから，1 秒あたり約 23.76 回の追跡座標の更新が行えることが分かった．また，PC の消費電力をワットメータ（REVEX ET30D）を用いて計測したところ，約 36W であった．これらから，PC では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 0.66 回の追跡座標の更新が行えることが分かった．

Fig. 5.41 は負荷状態の PC で人物追跡実行時の CPU の使用率である．4 スレッドの平均で 80% 程度の使用率が確認された．

人物追跡のソフトウェア・ハードウェア複合実装

ソフトウェア実装した人物追跡を ZedBoard の組み込み CPU（PS）と FPGA（PL）を用いてソフトウェア・ハードウェア複合実装した．本実験でソフトウェア・ハードウェア複合実装したのは，Fig. 5.34 の中と右に示す検証，実行モードである．この中で，ソフトウェア実装したのは白色部分，ハードウェア実装したのは橙色部分である．ハードウェア実装に関しては，人物検出と同様の専用アルゴリズムを流用している．

Table 5.12 にハードウェア実装のレイテンシ，Table 5.13 にハードウェア実装の資源使用率を示す．

Table 5.12　ハードウェア実装のレイテンシ

Latency	Min	Max
[ns]	257390	564590

Table 5.13　ハードウェア実装の資源使用率

Resource	Available	Used	Untilization
BRAM_18K	280	0	0%
DSP_48E	220	0	0%
FF	106400	8970	8%
LUT	53200	16942	31%

Table 5.14 ソフトウェア・ハードウェア複合実装した人物追跡の認識結果

		Input	
		Positive	Negative
Predict	Positive	194	7
	Negative	6	1243

Table. 5.14 にソフトウェア・ハードウェア複合実装した人物追跡の認識結果をまとめる．これは，検証モードに Positive（人物）画像を 200 枚，Negative（非人物）画像を 1250 枚入力し，認識結果をまとめたものである．Positive 画像を入力し，Poisitive 画像と認識されたのは，194 枚であった（TP）．反対に，Positive 画像を入力し，Negative 画像と認識されたのは，6 枚であった（FN）．また，Negative 画像を入力し，Negative 画像と認識されたのは，1243 枚であった（TN）．反対に，Negative 画像を入力し，Positive 画像と認識されたのは，7 枚であった（FP）．これらの値から正解率を式 5.1 から求めると，99.10% となった．

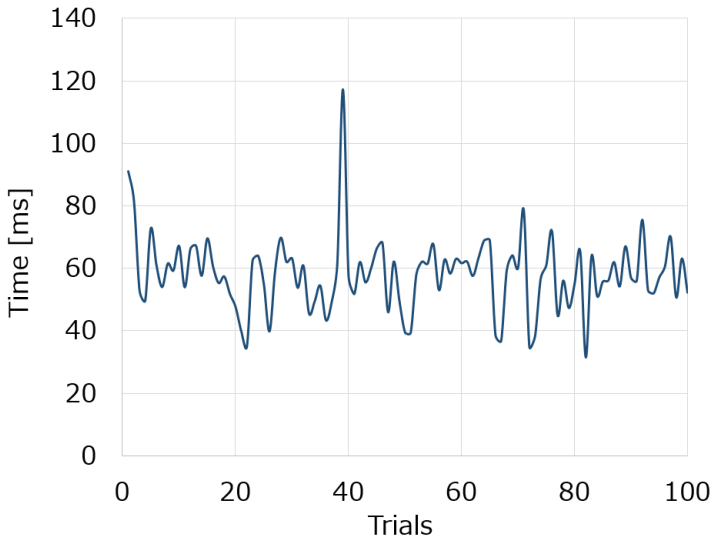


Fig. 5.42　組み込み CPU・FPGA による人物追跡処理速度

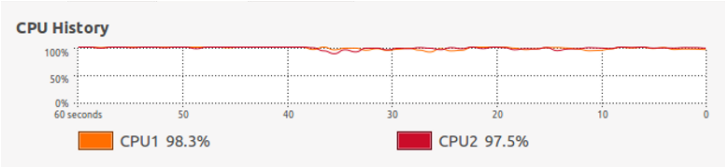


Fig. 5.43　組み込み CPU・FPGA に対する人物追跡の負荷

Fig. 5.42 はソフトウェア・ハードウェア複合実装した人物追跡を ZedBoard の組み込み CPU（PS）・FPGA（PL）で実行した時の処理速度である．これは，実行モードで画像入力から追跡座標の出力までに要した時間（1 回の追跡座標の更新に要した時間）をまとめたものである．グラフの縦軸は，1 回の追跡座標の更新に要した時間，横軸は試行回数で，1000 回試行した結果を示す．

1 回の追跡座標の更新に要した時間の平均は 57.96ms で，分散は 140.68ms であった．このことから，1 秒あたり約 17.25 回の追跡座標の更新が行えることが分かった．また，ZedBoard の消費電力を計測したところ，約 4.68W であった．これらから，組み込み CPU・FPGA では，単位電力あたり（1W あたり），単位時間（1 秒間）に約 3.69 回の追跡座標の更新が行えることが分かった．

Fig. 5.43 は組み込み CPU・FPGA で人物追跡実行時の CPU の使用率である．2 コア共に 100% 程度の使用率が確認された．

Table 5.15 人物追跡の結果まとめ

	UPS [ups]	UPS Per Watt [upspw]
PC（No Load）	29.06	1.12
PC（Load）	23.76	0.66
ZedBoard（CPU + FPGA）	17.25	3.69

この実験の結果を Table 5.15 にまとめる．ここで，UPS（The number of Update Per Second）とは 1 秒あたりの追跡座標の更新を指す．この表より，単位電力あたりの計算性能を評価すると，ZedBoard の組み込み CPU（PS）· FPGA（PL）でソフトウェア・ハードウェア複合実装した結果が最も良いことが分かる．

第 6 章

SoC 開発環境と ROS の統合

本章では，前章で述べた，ロボットミドルウェアとアクセラレータの統合法を改良し，SoC 開発環境と ROS を統合することで，システムの設計自由度の向上を狙う．

6.1 提案手法

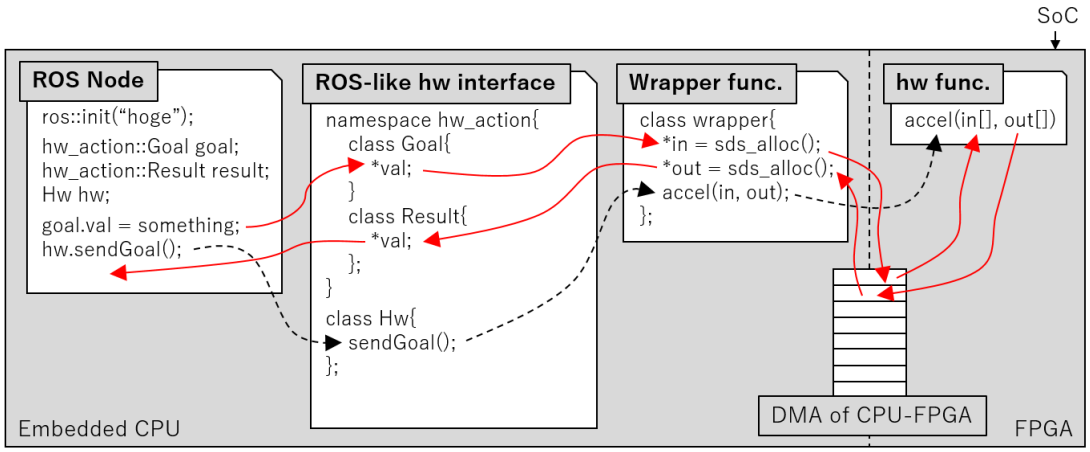


Fig. 6.1　ホームサービスロボットのための知的処理アクセラレータ

本研究では，ホームサービスロボットのための知的処理アクセラレータを提案する．提案手法は，hw/sw 複合体として SoC の Xilinx Zynq を用い，ROS インタフェースと互換性がある hw インタフェースを，sw に提供する．以下に，提案手法を用い，知的処理アクセラレータを統合したシステムを構築する手法を述べる．

(1) 回路技術者が，高位合成ツール Xilinx Vivado HLS を用いて，知的処理のハードウェア関数を記述する．

(2) 提案手法が，ハードウェア関数のディレクティブおよび引数を参照し，ハードウェア関数のソフトウェアラッパー関数を自動生成する．また，SoC の開発環境である Xilinx SDSoC を用いて，ハードウェア関数及びソフトウェアラッパー関数をビルドする．この時，ハードウェア関数は FPGA のコンフィギュレーション情報である bitstream とし，ソフトウェアラッパー関数は共有ライブラリとし出力する．

(3) 提案手法が，ソフトウェアラッパー関数を参照し，ROS インタフェースと互換性がある，ROS-like hw interface を自動生成する．この時，SDSoC でビルドされたソフトウェアラッパー関数は，同期動作をとるため，ROS action interface と互換性を持たせる．

(4) ロボット技術者が，ROS ノードを記述する．この時，ROS ノード内で ROS-like hw interface を用いることで，ROS action interface と適合したコード記述で知的処理を使用できる．

6.2　実験・考察

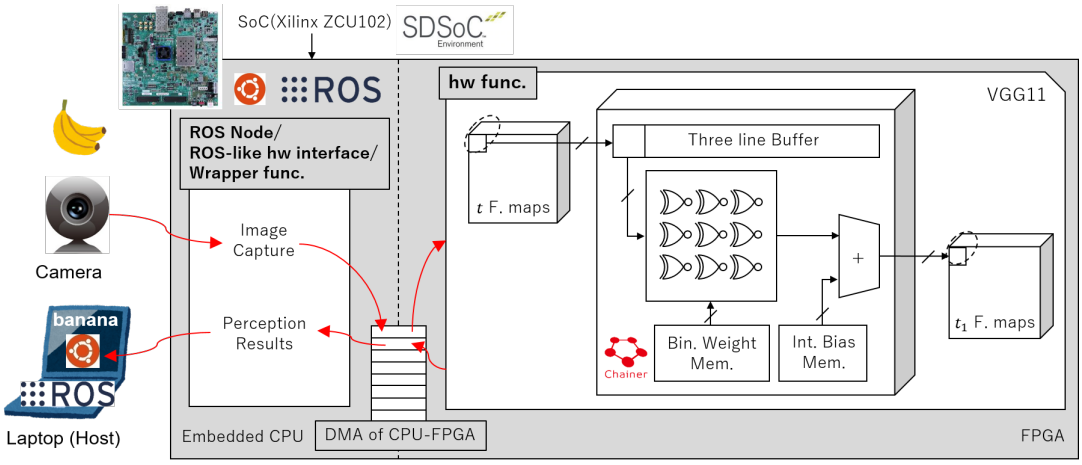


Fig. 6.2　アクセラレータの実験

提案手法を用いて，知的処理アクセラレータを統合したシステムを構築し，他の手法と処理速度を比較した．以下に，提案手法を用いたシステムの構築方法を述べる．

(1) Yonekawa らの手法 [69] を参考に，Chainer を用いて，VGG11 の重みを 2 値化した CNN（Convolutional Neural Network）を実装した．そして，日用品・雑貨・家事道具などが撮影された Washington RGB-D v1 データセットの RGB 画像を用いて CNN を訓練した．この時，51 カテゴリの物体に対して，訓練には 155816 枚，テストには 52104 枚の画像を用いた．
(2) Vivado HLS を用いて 2 値化した CNN のハードウェア関数を実装した．また，前述の訓練で得た重みを FPGA の BRAM に実装した．
(3) 提案手法を用いて，ソフトウェアラッパー関数，ROS-like hw interface を自動生成した．このインターフェースは，hw に RGB 画像を送り，hw から CNN によるカテゴリ分類結果を受け取る．
(4) ROS-like interface を用いて ROS ノードを記述した．このノードは，sensor_msgs/Image 型のメッセージをサブスクライブし，画像リサイズ後に ROS-like interface を介して CNN のハードウェア関数を実行し，カテゴリ分類の結果を std_msgs/String 型メッセージでパブリッシュする．

なお，本実験では，hw/sw 複合体として，Xilinx ZCU102 を用いた．

Table 6.1 2 値化 CNN の計算性能の比較, CPU: Intel Core i7 7820HK 3.9GHz GPU: NVIDIA GTX 1080

	CPU	GPU	Proposed
Frame rate [fps]	12.5	105.4(30.0)	29.8
Power [W]	48.4	70.0(38.6)	25.0
Efficiency [fps/W]	0.26	1.50(0.78)	1.19

提案手法，CPU，GPU による 2 値化した CNN の処理速度を比較した結果を Table. 6.1 に示す．この結果より，最も高速に処理が可能なデバイスは GPU であることが分かる．また，最も消費電力が低い計算方法は，提案手法を用いたシステムであることが分かる．最後に，単位電力当たりのフレームレート，すなわち計算効率は，GPU が最も良いことが分かる．しかしながら，ホームサービスロボットに搭載

される RGB-D カメラのフレームレートは 30fps であり，GPU の処理速度には余剰がある．ここで，GPU の処理速度を 30fps まで下げた場合の結果を Table. 6.1 の括弧内に示す．この場合，提案手法の計算効率が GPU を抜き，最も良い結果となった．

第 7 章

結論

7.1 まとめ

7.1.1 点群処理による物体検出・DNN による物体認識

ホームロボットにおいて重要な要素である物体認識・把持を行うため，3 次元点群処理による物体検出，Deep Learning による物体認識，マーカーベースのビジュアルフィードバック制御を組み合わせたシステムを提案し，構築，評価した．Deep Learning による物体認識では，3 方式の評価を行い，処理速度は遅いが 99% の物体認識率を有する方式を採用した．また，ビジュアルフィードバック制御を先行研究の色空間ベースからマーカーベースに変更することにより，物体や背景に影響を受けることがなくなり，物体把持成功率を 21 ポイント向上することに成功した．

7.1.2 DNN による end-to-end の物体検出・認識

本研究では，ホームサービスロボットの物体検出・認識のための半自動データセット作成法を提案した．実験の結果，提案手法は人手によるデータセット作成に比べ，1 つのデータ作成に要する作業時間を，167s から 0.58s へ約 287 分の 1 に短縮した．また，提案手法で作成したデータセットで DNN を訓練し，それをホームサービスロボットへ実装した．その結果，ロボットは全 10 回のうち，物体に触れることに 8 回成功し，物体の把持に 4 回成功した．

7.1.3　高位合成と ROS の統合

本研究では，ロボットミドルウェアとアクセラレータである FPGA を統合する“COMTA”（Connective Object for Middleware To Accelerator）を提案した．COMTA は特に，ロボットミドルウェアのインタフェースと FPGA 技術者が実装した知的処理を自動的に統合し，自由にシステムを再構成できることが特徴である．これは，ロボット技術者，FPGA 技術者双方にとって効率的な開発が可能となり，異分野の導入障壁がないシステムである．これにより，ロボット組み込み環境において，FPGA を用いた知的処理アクセラレータが急速に浸透していくと考えられる．

また，本研究では COMTA を用いた基礎実験を行った．この結果，各種通信，デバイスとのデータ交換において，知的処理に対応可能な性能を有していることが確認できた．特に，ROS のインタフェースからシステムの再構成を確認した実験は，前述の導入障壁が取り除かれたことを示す結果となっている．

また，システムをホームサービスロボットや自動運転車に搭載することを見据えた実験を行った．この実験では，画像による人物検出，追跡を FPGA に実装する対象とし，システムの評価を行った．結果として，従来の PC のみのシステムと比べ，高効率なシステムであることが分かった．

7.1.4　SoC 開発環境と ROS の統合

本研究では，知的処理アクセラレータをホームサービスロボットへ統合する手法を提案した．この結果，高位合成ツールで実装された知的処理ハードウェア関数に対し，ROS インタフェースと互換性を持ったインタフェースを自動生成できた．これにより，回路技術者はハードウェア関数の実装のみに専念でき，ロボット技術者は ROS と互換性を持ったインタフェースでハードウェア関数を利用できるようになった．

7.1.5　本論文のまとめ

本研究では，ホームサービスロボットのための基盤システムの確立を行った．これにより，今後ホームサービスロボットの研究開発の加速が図れる．また，プラットフォームの一部として，物体検出・認識および知的処理アクセラレータの構築を行っ

た．これにより，ホームサービスロボットへ DNN を応用する道筋をたて，またその効率的な計算機の設計手法を確立した．

7.2 今後の課題

今後の課題を以下にまとめる．

- DNN による end-to-end の物体検出・認識の提案手法をホームサービスロボットへ実装する時，物体操作の成功回数を向上する必要がある
- DNN による end-to-end の物体検出・認識を FPGA に実装する．
- 画像処理のみならず，様々な機械学習などの知的処理を実装する

最初の課題は，物体にわずかに触れている場合（Fig. 4.7（7）（8））, および物体に触れていない場合（Fig. 4.7（9）（10））について，物体領域が正しく推論されているか検証する必要がある．また，物体に触れたが持ち上げられない場合（Fig. 4.8（5）（6））について，物体領域だけでなく，ロボットが把持すべき点も教師したデータセットを生成することが考えられる．

次の課題は，第 4 章でロボット実機で動作を確認した，DNN による end-to-end の物体検出・認識を FPGA に実装出来ていないことである．このため，FPGA 実装を行い，ロボット実機での評価実験を行い，他の計算機と精度・処理速度・電力効率を比較する必要がある．

最後の課題として，今後は機械学習など様々な知的処理を実装する必要性がある．本研究では，画像の知的処理の効率化を実験した．しかし，ロボットに搭載される知的処理はこれに限られないため，今後は応用例を増やし，有効性の確認を行う必要がある．

謝辞

本研究を行うにあたり，非常に多くのご指導をいただくとともに，様々な議論を交わし，研究上大切な多くの経験，充実した研究環境，様々なアイデアを授けてくださった九州工業大学　大学院　生命体工学研究科　人間知能システム工学専攻の田向権准教授に心より感謝いたします．

ホームサービスロボットの開発に多くのアドバイスをいただくとともに様々な議論を交わしました，九州工業大学　大学院　生命体工学研究科　人間知能システム工学専攻の森江隆教授に心より感謝いたします．

同じ研究分野の研究者として，貴重なご意見や様々な知見をご教授下さった，九州工業大学　大学院　生命体工学研究科　人間知能システム工学専攻の堀尾恵一教授，東海大学　高輪校舎 情報通信学部組込みソフトウェア工学科 大川猛准教授に深く感謝申し上げます．

有志として集まり，様々な苦楽を含めホームサービスロボットの開発を共にしました，Hibikino-Musashi@Home の皆様に心より感謝いたします．また，Hibikino-Musashi@Home の活動をご支援いただいた，九州工業大学 カーロボ連携大学院推進室様，同大学学生支援係の皆様に心より感謝いたします．

最後に，日頃から様々な意見やアドバイスを頂くとともに，良き相談相手として支えてくださった九州工業大学 大学院 生命体工学研究科 人間知能システム工学専攻の田向研究室の皆様に心より感謝いたします．

参考文献

[1] 経済産業省, NEDO, ロボット産業将来市場調査, http://www.nedo.go.jp/news/press/AA5_0095A.html, 2019 年 11 月 1 日参照

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," IEEE International Conference on Robotics and Automation Workshop on Open Source Software, 2009.

[3] 岡田 慧, "ROS, 5 年経って, " 日本ロボット学会誌, Vol.35, No.4, pp.270-273, 2017.

[4] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," IEEE/ACM International Conference on Cyber-Physical Systems, 2018.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[6] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features," European Conference on Computer Vision, pp.404-417, 2006.

[7] Y. Freund and R.E. Schapire, "A decisiontheoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, Vol.55, Issue 1, pp.119-139, 1997.

[8] J.A.K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," Neural Processing Letters, Vol.9, Issue 3, pp.293-300, 1999.

[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," International Conference on Neural

Information Processing Systems, Vol.1, pp.1097-1105, 2012.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions,” IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[11] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” International Conference on Learning Representations, 2015.

[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. C. Berg, “SSD: Single Shot MultiBox Detector,” European Conference on Computer Vision, pp. 21-37, 2016.

[14] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei, “ImageNet : A large-scale hierarchical image database,” IEEE Conference Computer Vision and Pattern Recognition, 2009.

[15] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” International Journal of Computer Vision, Vol.88, pp.303-338, 2010.

[16] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” European Conference on Computer Vision, 2014.

[17] ROS Wiki, `http://wiki.ros.org/`, 2019 年 11 月 1 日参照

[18] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W. Yoon, “RT-middleware: distributed component middleware for RT (robot technology),” IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3555-3560, 2015.

[19] OpenRTM-aist official website, `http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website`, 2019 年 11 月 1 日参照

[20] V-Sido OS, `https://www.asratec.co.jp/v-sido-os/`, 2019 年 11 月 1 日参照

[21] Willow Garage, `http://www.willowgarage.com/`, 2019 年 11 月 1 日参照

[22] Open Source Robotics Foundation, https://www.osrfoundation.org/, 2019 年 11 月 1 日参照

[23] ROS Community Metrics Report 2019, http://download.ros.org/downloads/metrics/metrics-report-2019-07.pdf, 2019 年 11 月 1 日参照

[24] S. Hori, I. Yutaro, Y. Kiyama, Y. Tanaka, Y. Kuroda, M. Hisano, Y. Imamura, T. Himaki, Y. Yoshimoto, Y. Aratani, K. Hashimoto, G. Iwamoto, H. Fujita, T. Morie and H. Tamukoh, "Hibikino-Musashi@Home 2017 Team Description Paper," arXiv:1711.05457, 2017.

[25] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara and K. Murase, "Development of Human Support Robot as the research platform of a domestic mobile manipulator," ROBOMECH Journal, 2019.

[26] M. Wise, M. Ferguson, D. King, E. Diehr and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," Workshop on Autonomous Mobile Service Robots, 2016.

[27] P. Jordi, M. Luca and F. Francesco, "TIAGo: the modular robot that adapts to different research needs," International Workshop on Robot Modularity, 2016.

[28] T. Wisspeintner, T. van der Zant, L. Iocchi and S. Schiffer, "RoboCup Home: Scientific Competition and Benchmarking for Domestic Service Robots," Interaction Studies, pp. 392-426, 2009.

[29] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura and T. van der Zant, "Analysis and results of evolving competitions for domestic and service robots," Artificial Intelligence, pp. 258-281, 2015.

[30] H. Okada, T. Inamura and K. Wada, "What competitions were conducted in the service categories of the World Robot Summit?," Advanced Robotics, 2019.

[31] World Robot Challenge Partner Robot Challenge Real Space Rules & Regulations, https://worldrobotsummit.org/download/rulebook-en/rulebook-Partner_Robot_Challenge.pdf, 2019 年 11 月 1 日参照

[32] B. Khaled, A. Ali, L. Cheng, S. Yang, L. Ying and T. Xiang, "High performance biological pairwise sequence alignment: FPGA versus GPU versus cell BE versus GPP," International Journal of Reconfigurable Computing,

Vol. 2012, pp. 1-15, 2012.

[33] 田向 権, 黒木 良介, 華井 健太郎, 渡辺 雅史, 松下 宗一郎, 小林 祐一, 関根 優年, “インターネットブースター：ネットワーク配信可能な hw/sw 複合体を用いた WEB アプリケーション,” 電子情報通信学会論文誌, Vol. j93-D, No. 10, pp. 2139-2147, 2010.

[34] P. Hsiao, S. Lin and S. Huang, “An FPGA based human detection sysytem with embedded platform,” Microelectronic Enginnering, Vol. 138, pp. 42-46, 2015.

[35] L. Hofer, M. Tanaka, H. Tamukoh, A. F. Nassiraei and Takashi Morie, “Depth-Based Visual Servoing Using Low-Accurate Arm,” Joint 8th International Conference on Soft Computing and Intelligent Systems and 17th International Symposium on Advanced Intelligent Systems, 2016.

[36] OpenCV, `http://opencv.jp/`, 2019 年 11 月 1 日参照

[37] PCL, `http://pointclouds.org/`, 2019 年 11 月 1 日参照

[38] ar_track_alvar, `http://wiki.ros.org/ar_track_alvar`, 2019 年 11 月 1 日参照

[39] Caffe, `http://caffe.berkeleyvision.org`, 2019 年 11 月 1 日参照

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going Deeper with Convolutions,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1-9, 2015.

[41] ImageNet Large Scale Visual Recognition Challenge 2012, `http://image-net.org/challenges/LSVRC/2012/`, 2019 年 11 月 1 日参照

[42] Model Zoo, `http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel`, 2019 年 11 月 1 日参照

[43] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang and Y. Zhou, “Deep Learning Scaling is Predictable, Empirically,” arXiv:1712.00409, 2017.

[44] G. Georgakis, A. Mousavian, A. C. Berg and J. Kosecka, “Synthesizing Training Data for Object Detection in Indoor Scenes,” arXiv:1702.07836, 2017.

[45] G. Georgakis, Md. Reza, A. Mousavian, P. Le and J. Kosecka, “Multiview RGB-D dataset for object instance detection,” IEEE International Confer-

ence on 3DVision, 2016.

[46] K. Lai, L. Bo and D. Fox, “Unsupervised feature learning for 3D scene labeling,” IEEE International Conference on on Robotics and Automation, 2014.

[47] A. Mousavian, H. Pirsiavash and J. Kosecka, “Joint semantic segmentation and depth estimation with deep convolutional networks,” IEEE International Conference on 3DVision, 2016.

[48] C. Taylor and A. Cowley, “Parsing indoor scenes using RGB-D imagery,” Robotics: Science and Systems, 2012.

[49] A. Singh, J. Sha, K. Narayan, T. Achim and P. Abbeel, “A large-scale 3D database of object instances,” IEEE International Conference on Robotics and Automation, 2014.

[50] K. Lai, L. Bo, X. Ren and D. Fox, “A large-scale hierarchical multi-view RGB-D object dataset,” IEEE International Conference on on Robotics and Automation, 2011.

[51] Y. Boykov, O. Veksler and R. Zabih, “ Fast approximate energy minimization via graph cuts,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, Issue 11, pp. 1222-1239, 2001.

[52] M. Tanaka, R. Kamio and M. Okutomi, “Seamless image cloning by a closed form solution of a modified poisson problem,” Proceedings of Special Interest Group on Computer GRAPHics Asia, 2012.

[53] colorcorrect, `https://github.com/shunsukeaihara/colorcorrect`, 2019年11月1日参照

[54] A. B. Lange, U. P. Schultz and A. S. Soerensen, “Towards Automatic Migration of ROS Components from Software to Hardware,” 4th International Workshop on Domain-Specific Languages and Models for Robotic Systems, arXiv:1407.7560, 2014.

[55] A. B. Lange, U. P. Schultz and A. S. Soerensen, “Unity-link: A Software-Gateware Interface for Rapid Prototyping of Experimental Robot Controllers on FPGAs,” IEEE/RSJ International Conference on Intelligent Robots and Systems，pp. 3899-3906, 2013.

[56] T. Ohkawa, K. Yamashina, H. Kimura, K. Ootsu and T. Yokota, “FPGA

Components for Integrating FPGAs into Robot Systems," IEICE Transactions on Information and Systems, Vol.E101-D, No.2, pp.363-375, 2018.

[57] T. Ohkawa, K. Yamashina, T. Matsumoto, K. Ootsu and T. Yokota, "Automatic Generation Tool of FPGA Components for Robots," IEICE Transactions on Information and Systems, Vol.E102-D, No.5, pp.1012-1019, 2019.

[58] Xilinx, https://www.xilinx.com/, 2019 年 11 月 1 日参照

[59] ZedBoard, http://zedboard.org/, 2019 年 11 月 1 日参照

[60] Zynq-7000 All Programmable SoC https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html, 2019 年 11 月 1 日参照

[61] Xillybus, http://xillybus.com/, 2019 年 11 月 1 日参照

[62] OpenNI, http://www.openni.ru/, 2019 年 11 月 1 日参照

[63] Video for Linux, http://www.exploits.org/v4l/, 2019 年 11 月 1 日参照

[64] S.Iwata and S.Enokida, "Detection Based on Multiresolution Co-HOG," Advances in Visual Computing Lecture Notes in Computer Science, vol.8888, pp.427-437, 2014.

[65] INRIA Person Dataset, http://pascal.inrialpes.fr/data/human/, 2019 年 11 月 1 日参照

[66] INRIA Person Dataset, http://pascal.inrialpes.fr/data/human/, 2019 年 11 月 1 日参照

[67] Vivado High-Level Synthesis, https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, 2019 年 11 月 1 日参照

[68] Vivado Design Suite, https://www.xilinx.com/products/design-tools/vivado.html, 2019 年 11 月 1 日参照

[69] H. Yonekawa and H. Nakahara, "On-Chip Memory Based Binarized Convolutional Deep Neural Network Applying Batch Normalization Free Technique on an FPGA," Proceedings the IEEE International Parallel and Distributed Processing Symposium Workshops, pp.98-105, 2017.

研究実績

論文誌（筆頭著者，査読有）

- Yutaro Ishida and Hakaru Tamukoh, “Semi-automatic Dataset Generation for Object Detection and Recognition and its Evaluation on Domestic Service Robots,” Journal of Robotics and Mechatronics, Vol. 32, No. 1, pp. 245-253, 2020.
- Yutaro Ishida, Takashi Morie and Hakaru Tamukoh, “A Hardware Intelligent Processing Accelerator for Domestic Service Robots,” Advanced Robotics. (Submitted)

学会発表（筆頭著者，査読有）

- ○ Yutaro Ishida, Hakaru Tamukoh, “High-Level Synthesis System to Integrate SoC and ROS,” Asia Pacific Conference on Robot IoT System Development and Platform, Pattaya, Thailand, November 1-4(2), 2019.
- ○石田 裕太郎，“Hibikino-Musashi@Home: ホームサービスロボット開発学生プロジェクトの紹介，” ROSCon JP 2018，2019 年 9 月 15 日，東京，TKP ガーデンシティ PREMIUM 秋葉原.
- ○ Yutaro Ishida, Takashi Morie, Hakaru Tamukoh, “Live Demonstration: A Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots,” IEEE International Symposium on Circuits and Systems, C2P-U, Florence, Italy, May 27-30(30), 2018.
- ○ Yutaro Ishida, Takashi Morie, Hakaru Tamukoh, “A Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots,”

IEEE International Symposium on Circuits and Systems, A2P-0, Florence, Italy, May 27-30(27), 2018.

学会発表（筆頭著者，査読無）

- ○石田 裕太郎, 安藤 充宏, 能勢 啓輔, 田向 権，“ホームサービスロボットにおける命令文中の単語意味ベクトルに注目した最適行動識別，” ロボティクス・メカトロニクス講演会 2018，2P1-B18，2018 年 6 月 2-5 日 (5)，福岡，西日本総合展示場.
- ○ Yutaro Ishida, Yuichiro Tanaka, Sansei Hori, Yuta Kiyama, Yuki Kuroda, Masataka Hisano, Hiroto Fujita, Yuma Yoshimoto, Yoshiya Aratani, Goki Iwamoto, Kohei Hashimoto, Dinda Pramanta, Yushi Abe, Takashi Morie and Hakaru Tamukoh, “Approach to accelerate the development of practical home service robots -RoboCup@Home DSPL-,” 26th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN Workshop: HRI for Service Robots in RoboCup@Home, August 28-September 1(September 1), Lisbon, Pestana Palace Hotel, 2017. (Invited Talk)
- ○石田 裕太郎，堀 三晟，木山 雄太，黒田 裕貴，田中 悠一朗，久野 昌隆，吉元 裕真，今村 有杜，日巻 智貴，新谷 嘉也，岩元 剛毅，橋本 康平，森江 隆，田向 権，“Hibikino-Musashi@Home におけるロボット開発，” 第 7 回インテリジェントホームロボティクス研究会，2017 年 6 月 9 日，大阪，インテックス大阪.（招待講演）
- ○石田 裕太郎，大川 猛，田向 権，“FPGA のためのロボットミドルウェアインタフェースの基礎検討，” 第 61 回システム制御情報学会研究発表講演会，324-2，2017 年 5 月 23-25 日 (25)，京都，京都テルサ.（SCI 学生発表賞）
- ○ Yutaro Ishida, Sansei Hori, Takashi Morie, Hakaru Tamukoh, “Evaluation of High Speed Processing System for Service Robots,” The 3rd Int. Conf. on Computational Methods in Engineering and Health Sciences (ICCMEH2016), P-2 pp. 34, Kyushu Institute of Technology Tobata Campus, Fukuoka, December 17-18(17), 2016.

- ○石田 裕太郎，田中 悠一朗，森江 隆，田向 権，“ホームロボット向け物体認識・把持システムの構築，” 第 34 回日本ロボット学会学術講演会，3G1-03，2016 年 9 月 7-9 日 (9)，山形，山形大学小白川キャンパス．
- ○石田 裕太郎，橋本 康平, 有田 裕太，田中 良道，西田 健，伊藤 太久磨，井上 秀雄，通山 恭一, 田向 権，“自動運転のための知的処理におけるハードウェアアクセラレータ利用の基礎検討，” 第 32 回ファジィシステムシンポジウム，WD1-1, pp.49-50, 2016 年 8 月 31 日-9 月 2 日 (8 月 31 日)，佐賀，佐賀大学．（ポスター・デモセッション最優秀賞）
- ○石田 裕太郎，堀 三晟，森江 隆，田向 権，“FPGA による ROS 向け高速分散処理システムの実装，” 第 60 回システム制御情報学会研究発表講演会，111-5，2016 年 5 月 25-27 日 (25)，京都，京都テルサ．（学会賞奨励賞）
- ○石田 裕太郎，田中 宙夫，森江 隆，田向 権，“ホームロボットへの応用を目指した ROS と FPGA の連携システムの構築，” 第 33 回日本ロボット学会学術講演会，3F3-03，2015 年 9 月 3-5 日 (5)，東京，東京電機大学東京千住キャンパス．
- ○石田 裕太郎，田中 宙夫，森江 隆，田向 権，“ホームロボット向け組み込み高速演算システムの構築，” 日本知能情報ファジィ学会九州支部春季ワークショップ 2015，講演番号 18，2015 年 6 月 20-21 日 (21)，熊本，東海大学阿蘇キャンパス．
- ○石田 裕太郎，田中 宙夫，森江 隆，田向 権，“家庭用サービスロボットのための ROS-FPGA 間通信の検討，” 電子情報通信学会総合大会情報システムソサエティ特別企画学生ポスターセッション，p.201，2015 年 3 月 10-13 日 (12)，滋賀，立命館大学びわこ・くさつキャンパス．

論文誌（共著者，査読有）

- 橋本 康平，石田 裕太郎，市瀬 龍太郎，我妻 広明，田向 権，“論理知識型 AI に基づく自動運転のための危険予測システムの構築と評価，” システム制御情報学会論文誌，Vol.31，number 5，pp.191-201，2018．

学会発表（共著者，査読有）

- ○ Masatoshi Yamaguchi, Gouki Iwamoto, Yushi Abe, Yuichiro Tanaka, Yutaro Ishida, Hakaru Tamukoh and Takashi Morie, “Live Demonstration: a VLSI implementation of time-domain analog weighted-sum calculation model for intelligent processing on robots,” IEEE International Symposium on Circuit and Systems, Paper ID 2353, Sapporo, Japan, May 26-29(27), 2019. (Best Live Demonstration Award)
- ○ Takeshi Ohkawa, Yutaro Ishida, Yuhei Sugata and Hakaru Tamukoh, “ROS-Compliant FPGA Component Technology - FPGA installation into ROS,” ROSCon, Vancouver, Canada, September 21-22(22), 2017.

学会発表（共著者，査読無）

- ○阿部 佑志，石田 裕太郎，小野 智寛，田向 権，“物体の 3D スキャンによるホームサービスロボット向け学習データセット作製の高速化，” 第 37 回日本ロボット学会学術講演会，2K2-02，2019 年 9 月 3-7 日 (5)，東京，早稲田大学早稲田キャンパス.
- ○宮崎 椋瑚，三井 悠也，石田 裕太郎，渡辺 政彦，宇井 健一，横田 剛典，市瀬 龍太郎，我妻 広明，田向 権，“決定表に基づく自動運転用判断システムの構築と検証，” 第 35 回ファジィシステムシンポジウム講演会，SB3-3，2019 年 8 月 29-31 日 (31)，大阪，大阪大学.
- ○金岡 大樹，石田 裕太郎，田向 権，“ホームサービスロボットにおける触覚情報を用いた物体認識，” SOFT 九州支部夏季ワークショップ 2019，18，2019 年 8 月 22-23 日 (22)，福岡，福岡サンパレス.（学生優秀ポスター発表賞）
- ○小野 智寛，堀 三晟，石田 裕太郎，田中 悠一朗，吉元 裕真，阿部 佑志，武藤 冬樹，椛島 康平，福宿 将士，坂田 拓馬，吉井 拓巳，上村 大地，金丸 和樹，中村 健太郎，西村 雄太，森江 隆，田向 権，“社会実装を目指したホームサービスロボットの研究開発，” ロボティクス・メカトロニクス講演会 2019，1P2-102，2019 年 6 月 5-8 日 (6)，広島，広島国際会議場.
- ○岩渕 甲誠，松本 茂樹，松尾 幸治，石田 裕太郎，廣瀬 尚三，長瀬 雅之，田

向 権，“高位合成による ORB-SLAM の FPGA 実装と評価，” ロボティクス・メカトロニクス講演会 2019，1AI-F06，2019 年 6 月 5-8 日 (6)，広島，広島国際会議場.

- ○小野 智寛，石田 裕太郎，阿部 佑志，田向 権，“ホームサービスロボットにおける物体把持のための把持点推定，” 第 9 回インテリジェントホームロボティクス研究会，2019 年 3 月 24 日，大阪，大阪工業大学梅田キャンパス.
- ○三好 竜平，宮 椋瑚，橋本 康平，石田 裕太郎，渡辺 政彦，宇井 健一，市瀬 龍太郎，我妻 広明，田向 権，“自動運転のための運転行動意思決定システム開発に向けた統合シミュレータの構築，” 第 34 回ファジィシステムシンポジウム講演会，TH1-3，2018 年 9 月 3-5 日 (4)，愛知，名古屋大学.
- ○三好 竜平，宮 椋瑚，橋本 康平，石田 裕太郎，渡辺 政彦，宇井 健一，市瀬 龍太郎，我妻 広明，田向 権，“自動運転のための運転行動意思決定システム開発に向けた統合シミュレータの提案，” SOFT 九州支部夏季ワークショップ 2018，31，2018 年 8 月 28-29 日 (29)，鹿児島，霧島国際ホテル.（学生優秀ポスター発表賞）
- ○ Valentin Bilot，Yutaro Ishida，Patrick Henaff and Hakaru Tamukoh，“Grasping objects on the floor with home service robot，” SOFT 九州支部夏季ワークショップ 2018in 霧島，2018 年 8 月 28-29 日 (29)，鹿児島, 霧島国際ホテル.
- ○吉元 裕真，堀 三晟，石田 裕太郎，木山 雄太，黒田 裕貴，田中 悠一朗，久野 昌隆，藤田 啓斗，新谷 嘉也，岩元 剛毅，橋本 康平，森江 隆，田向 権，“競技会活動を通じたホームサービスロボットの研究開発，” ロボティクス・メカトロニクス講演会 2018，2P2-E02，2018 年 6 月 2-5 日 (5)，福岡，西日本総合展示場.
- ○岩渕 甲誠，内田 大貴，石田 裕太郎，長瀬 雅之，田向 権，“AP SoC による FPGA と RT ミドルウェアの連携，” ロボティクス・メカトロニクス講演会 2018，2A1-G10，2018 年 6 月 2-5 日 (5)，福岡，西日本総合展示場.
- ○田向 権，西田 健，石田 裕太郎，“高速化・省電力化が期待されるロボットの知的処理，” 電子情報通信学会リコンフィギャラブルシステム研究会，RECONF2018-16，2018 年 5 月 24-25 日 (25), 東京，ゲートシティ大崎.（依頼講演）
- ○三好 竜平，石田 裕太郎，橋本 康平，渡辺 政彦，宇井 健一，市瀬 龍太郎，

我妻 広明，田向 権，“オントロジーに基づく自動運転向け意思決定システムのAutoware への実装，” 電子情報通信学会総合大会情報システムソサエティ特別企画学生ポスターセッション，ISS-SP-060，2018 年 3 月 20-22 日 (21)，東京，東京電機大学東京千住キャンパス．

- ○橋本 康平，石田 裕太郎，三好 竜平，市瀬 龍太郎，我妻 広明，田向 権，“オントロジーに基づく自動運転向け意思決定システムの推論速度評価，” 第 33 回ファジィシステムシンポジウム講演会，FA2-3，pp.681-684，2017 年 9 月 13-15 日 (15)，山形，山形大学．（ポスター・デモセッション優秀賞，奨励賞）
- ○橋本 康平，石田 裕太郎，市瀬 龍太郎，我妻 広明，田向 権，“自動運転のための理論知識型 AI での危険予測における推論能力の基礎検討，” 第 61 回システム制御情報学会研究発表講演会，324-1，2017 年 5 月 23-25 日 (25)，京都，京都テルサ．（学会賞奨励賞）
- ○堀 三晟，石田 裕太郎，奥村 弘治，木山 雄太，楠根 穣，田中 悠一朗，辻 湧弥，土田 崇 弘，土谷 諒，藤本 武，山崎 裕太，佐藤 寧，森江 隆，田向 権，“Hibikino-Musashi@Home チームにおけるロボット開発，” 第 4 回インテリジェントホームロボティクス研究会，2016 年 5 月 27 日，大阪，インテックス大阪．（招待講演）

その他発表

- 柴田 智広，石田 裕太郎，藤田 啓斗，吉元 裕真，第 2 回 HSR ユーザ会，2017 年 2 月 28 日，愛知，トヨタ産業技術記念館．（優秀ポスタープレゼンテーション賞）
- ○石田 裕太郎，田向 権，“ROSCon2017 参加報告，” ROSCon 2017 勉強会，2017 年 10 月 23 日，早稲田大学．

特許

- 田向 権，石田 裕太郎，森江 隆，出願人: 九州工業大学，“ハードウェアプラットフォーム及びハードウェアプラットフォームを用いたハードウェアの操作方法，” 特願 2016-037660，特開 2017-156862，特許第 6611635 号．

受賞

- Hibikino-Musashi@Home，@ホーム・ドメスティックスタンダードプラットフォームリーグ優勝，令和元年 8 月 18 日，ロボカップジャパンオープン 2019 ながおか開催委員会.
- Hibikino-Musashi@Home，@ホーム・オープンプラットフォームリーグ優勝，令和元年 8 月 18 日，ロボカップジャパンオープン 2019 ながおか開催委員会.
- Hibikino-Musashi@Home，優勝（DSPL），令和元年 8 月 18 日，ロボカップジャパンオープン 2019 @ホームリーグ実行委員会.
- Hibikino-Musashi@Home，優勝（OPL），令和元年 8 月 18 日，ロボカップジャパンオープン 2019 @ホームリーグ実行委員会.
- Hibikino-Musashi@Home, RoboCup@Home League Domestic Standard Platform Third Place.
- Hibikino-Musashi@Home，最優秀賞，平成 31 年 3 月 1 日，平成 30 年度学生プロジェクト報告会.
- Hibikino-Musashi@Home, World Robot Summit 2018 Tokyo RSJ Special Award, October 21st 2018, The Robotics Society of Japan.
- Hibikino-Musashi@Home, World Robot Summit 2018 Tokyo Service Robotics Category, Partner Robot Challenge / Real Space 1st Place, METI Minister’s Award for Excellence in WRS, October 21st 2018, Ministry of Economy, Trade and Industry.
- Hibikino-Musashi@Home, RoboCup@Home P&G Challenge Winner, June 21st 2018, Procter & Gamble.
- Hibikino-Musashi@Home, RoboCup@Home League Procter & Gamble Dishwasher Challenge Award, June 21st 2018, RoboCup Federation, RoboCup 2018.
- Hibikino-Musashi@Home, RoboCup@Home League Domestic Standard Platform RoboCup@Home First Place, June 21st 2018, RoboCup Federation, RoboCup 2018.
- Hibikino-Musashi@Home DSPL，人工知能学会賞「少試行のインタラクションによるホームロボットの知能獲得」，平成 30 年 5 月 5 日，一般社団法人 人

工知能学会.

- Hibikino-Musashi@Home DSPL，ロボカップ@ホームドメスティックスタンダードプラットフォームリーグ第 2 位，平成 30 年 5 月 5 日，ロボカップジャパンオープン 2018 おおがき開催委員会.
- Hibikino-Musashi@Home OPL，ロボカップ@ホームオープンプラットフォームリーグ第 1 位， 平成 30 年 5 月 5 日，ロボカップジャパンオープン 2018 おおがき開催委員会.
- Hibikino-Musashi@Home DSPL，ロボカップジャパンオープン 2018 @ホームリーグ 準優勝（DSPL），平成 30 年 5 月 5 日，ロボカップジャパンオープン 2018 @ホームリーグ実行委員会.
- Hibikino-Musashi@Home OPL，ロボカップジャパンオープン 2018 @ホームリーグ 優勝（OPL），平成 30 年 5 月 5 日，ロボカップジャパンオープン 2018 @ホームリーグ実行委員会.
- Hibikino-Musashi@Home，RoboCup@Home League Domestic Standard Platform 1st Place, June 30th 2017, RoboCup Federation, RoboCup2017.
- Hibikino-Musashi@Home，RoboCup@Home Domestic Standard Platform 第 2 位，平成 29 年 5 月 5 日，ロボカップジャパンオープン 2017 開催委員会.
- Hibikino-Musashi@Home，RoboCup@Home Open Platform 第 3 位，平成 29 年 5 月 5 日，ロボカップジャパンオープン 2017 開催委員会．v
- Hibikino-Musashi@Home，ロボカップジャパンオープン 2017@Home Domestic Standard Platform 準優勝，平成 29 年 5 月 5 日，ロボカップジャパンオープン 2017 @ Home リーグ実行委員会.
- Hibikino-Musashi@Home，ロボカップジャパンオープン 2017@Home Open Platform 三位，平成 29 年 5 月 5 日，ロボカップジャパンオープン 2017 @ Home リーグ実行委員会.
- Hibikino-Musashi@Home，Intelligent Home Robotics Challenge 2016 総合優勝，平成 28 年 12 月 11 日，日本ロボット学会インテリジェントホームロボティクス研究専門委員会.
- Hibikino-Musashi@Home, Intelligent Home Robotics Challenge 2016 ロボットマニピュレーション部門 1 位，平成 28 年 12 月 11 日，日本ロボット学会インテリジェントホームロボティクス研究専門委員会.
- Hibikino-Musashi@Home, Intelligent Home Robotics Challenge 2016 ロボッ

ト聴覚部門 1 位，平成 28 年 12 月 11 日，日本ロボット学会インテリジェントホームロボティクス研究専門委員会.

- Hibikino-Musashi@Home，ロボカップ@ホームオープンプラットフォームリーグ準優勝，平成 28 年 3 月 27 日，ロボカップジャパンオープン 2016 愛知開催委員会.
- Hibikino-Musashi@Home，ロボカップ@ホームオープンプラットフォームリーグファイナリスト賞 3 位，平成 28 年 3 月 27 日，ロボカップジャパンオープン 2016 愛知開催委員会.
- Hibikino-Musashi@Home，ロボカップジャパンオープン 2016@Home Open Platform 準優勝，平成 28 年 3 月 27 日，ロボカップジャパンオープン 2016@Home リーグ実行委員会.
- 連携大学院インテリジェントカー・ロボティクスコース，2015 年 自動運転支援センシング技術総合実習 チーム Odd's，最優秀賞.
- Hibikino-Musashi@Home，ロボカップ@ホーム実機リーグ第 3 位，平成 27 年 5 月 4 日，ロボカップジャパンオープン 2015 福井開催委員会.

その他の活動

- 石田裕太郎（Hibikino-Musashi@Home)，「家庭用サービスロボットの実現に向けた競技会への参加とプラットフォームの提供」，九州工業大学創立 100 周年記念事業学生創造学習支援プロジェクト（平成 30 年度学生プロジェクト)，200 万円.
- 石田裕太郎（Hibikino-Musashi@Home)，「家庭用サービスロボットの実現に向けた競技会への参加とプラットフォームの提供」，九州工業大学創立 100 周年記念事業学生創造学習支援プロジェクト（平成 29 年度安川電機プロジェクト)，200 万円.
- 石田裕太郎（Hibikino-Musashi@Home)，「家庭用サービスロボットの実現に向けた競技会への参加とプラットフォームの提供」，九州工業大学創立 100 周年記念事業学生創造学習支援プロジェクト（平成 28 年度安川電機プロジェクト)，200 万円.